Exhibit A

```
'From Squeak 2.3 o
Object subclass: #AnimationObject
    instanceVariableNames: ''
    classVariableNames: 'ColorMapsDictionary InfiniteBounds '
    poolDictionaries: ''
    category: 'Numericon-Kernel'!
AnimationObject subclass: #AnimationElement
    instanceVariableNames: 'elements bounds location dirty isVisible isActive isEnabled isHilited step
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Numericon-Kernel'!
AnimationElement class
    instanceVariableNames: 'actions '!
AnimationElement subclass: #CounterAE
    instanceVariableNames: 'magnitude '
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Numericon-Counters'!
CounterAE subclass: #ClockAE
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Numericon-Counters'!
CounterAE subclass: #DigitAE
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Numericon-Counters'!
CounterAE subclass: #DigitCounterAE
    instanceVariableNames: 'minValue maxValue '
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Numericon-Counters'!
AnimationObject variableSubclass: #DisplayProfile
    instanceVariableNames: 'hash width height depth frameCount frameSequences '
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Numericon-Kernel'!
DisplayProfile class
    instanceVariableNames: ''!
AnimationObject subclass: #DisplayProfileFactory
    instanceVariableNames: 'profile '
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Numericon-Kernel'!
AnimationObject subclass: #FrameSequence
    instanceVariableNames: 'pointSteps formSteps points forms '
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Numericon-Kernel'!

!Object methodsFor: 'testing' stamp: 'm3r
isBoolean
    ^false! !

!Object methodsFor: 'testing' stamp: 'm3r :
isCollection
    ^false! !


!AnimationObject reorganize!
('utility functions' clip:by: does:contain: infiniteBounds merge:into: merge:into:clippedBy:)
('accessing color maps' colorMapForDepth:)
('private' initialize)
!


!AnimationObject methodsFor: 'utility functions' stamp: 'm3r
clip: dstArea by: clipArea
    clipArea == nil ifTrue: [^nil].
    dstArea == nil ifTrue: [^nil].
    dstArea intersectIntoSelf: clipArea.
    ^dstArea hasPositiveExtent ifTrue: [dstArea] ifFalse: [nil]
! !

!AnimationObject methodsFor: 'utility functions' stamp: 'm3r z
does: areaA contain: areaB
    areaA == nil ifTrue: [^false].
    areaB == nil ifTrue: [^true].
    ^areaA containsRect: areaB! !

!AnimationObject methodsFor: 'utility functions' stamp: 'm3r
infiniteBounds
    ^self class infiniteBounds
! !

!AnimationObject methodsFor: 'utility functions' stamp: 'm3r
merge: srcArea into: dstArea
    srcArea == nil ifTrue: [^dstArea].
    dstArea == nil ifTrue: [^srcArea copy].
    ^dstArea mergeIntoSelf: srcArea! !

!AnimationObject methodsFor: 'utility functions' stamp: 'm3r ;
merge: srcArea into: dstArea clippedBy: clipArea
    "dstArea is assumed to already be clipped."
    clipArea == nil ifTrue: [^nil].
    srcArea == nil ifTrue: [^dstArea].
    dstArea == nil ifTrue: [^srcArea copy intersectIntoSelf: clipArea].
    dstArea mergeIntoSelf: srcArea clippedBy: clipArea.
    ^dstArea hasPositiveExtent ifTrue: [dstArea] ifFalse: [nil]
! !

!AnimationObject methodsFor: 'accessing color maps' stamp: 'm3r
colorMapForDepth: depth
    ^self colorMaps at:
        (depth < 8 ifTrue: [depth] ifFalse: [7 - (depth bitShift: -3)])! !

!AnimationObject methodsFor: 'private' stamp: 'm3r ;
initialize
    self class ensure.! !


!AnimationElement reorganize!
('structure' addSubelement: context removeSubelement:ifAbsent: subelements subelements:)
('enumerating' allSubelementsDo: subelementsDo: subelementsReverseDo: withAllSubelementsDo: withSubel
('testing' containsActive containsVisible isActive isCompletelyDirty isCounter isDirty isEnabled isHil
('display profiling' depth depth: ensureDisplayProfile frameCount frameCount: hvFillRatio hvFillRatio:
('attributes setting' beActive beDisabled beEnabled beHilited beInactive beInvisible beUnhilited beVis
('accessing' backgroundImage backgroundImage: color color: creator creator: id id: step step:)
('private accessing' selfActivity selfActivity: selfEnable selfEnable: selfHilite selfHilite: selfSte
('properties' findPropertyAt:ifAbsent: findPropertyAt:ifAbsentAtTopContextPut: findPropertyAt:ifAbsen
('displaying' graphics internalExtent offset)
('drawing' drawOn:)
('actions' actionAt: actionAt:ifAbsent: actionAt:ifPresent: actionAt:ifPresent:ifAbsent:)
('change notification' aspect:update: aspect:update:with: aspectChanged: aspectChanged:with: sharedAsp
('handling change' activityChangedTo: aspectChanged basicAspectChanged boundsChanged coordinateSystem
('area calculation' coveredArea dirtyArea drawingArea maxDrawingArea maxVisibleArea validArea visibleA
('private area calculation' areaWithinContext basicDrawingArea basicVisibleArea mergeCoveredAreaWith:
('marking dirty' markAreaDirty: markBoundsDirty markClean markDirty markUncoveredAreaDirtyInContext m
('marking all dirty' markAllBoundsDirty markAllClean markAllDirty)
('absolute spatial accessing' absoluteBounds absoluteLocation absoluteOrigin convertPointToAbsolute:)
('external spatial accessing' convertAreaFromContext: convertAreaToContext: location location: locati
('internal spatial accessing' bounds bounds: boundsOffset extent extent:moveCoordinateSystemBy: moveCoordinat
('private spatial methods' getBounds setBounds: setLocation: setLocationBounds: setOriginOffsetBy:)
('private' initialize setExtent:depth: setGraphics:bounds:)
('defaults' defaultActivity defaultBackgroundImage defaultColor defaultDepth defaultEnable defaultHil
```

```
!

!AnimationElement methodsFor: 'structure' stamp: 'm3r
addSubelement: anElement
    elements _ elements isCollection
        ifFalse: [OrderedCollection with: elements]
        ifTrue: [elements asOrderedCollection].
    anElement setContext: self.
    ^elements add: anElement.! !

!AnimationElement methodsFor: 'structure' stamp: 'm3r
context
    ^elements isCollection
        ifFalse: [elements]
        ifTrue: [elements at: 1]
! !

!AnimationElement methodsFor: 'structure' stamp: 'm3r
removeSubelement: anElement ifAbsent: exceptionAction
    elements isCollection
        ifFalse: [^exceptionAction value]
        ifTrue: [elements remove: anElement ifAbsent: [^exceptionAction value]].
    anElement setContext: nil.
    ^anElement! !

!AnimationElement methodsFor: 'structure' stamp: 'm3r 2
subelements
    ^elements isCollection
        ifFalse: [#()]
        ifTrue: [elements allButFirst]
! !

!AnimationElement methodsFor: 'structure' stamp: 'm3r ;
subelements: aSequenceable
    aSequenceable do: [:element ! element setContext: self].
    ^elements _ aSequenceable copyWithFirst: self context
! !

!AnimationElement methodsFor: 'enumerating' stamp: 'm3r
allSubelementsDo: aBlock
    self subelementsDo:
        [:element ! element withAllSubelementsDo: aBlock].
! !

!AnimationElement methodsFor: 'enumerating' stamp: 'm3r ;
subelementsDo: aBlock
    elements isCollection ifTrue:
        [2 to: elements size do:
            [:index ! aBlock value: (elements at: index)]].
! !

!AnimationElement methodsFor: 'enumerating' stamp: 'm3r
subelementsReverseDo: aBlock
    elements isCollection ifTrue:
        [elements size to: 2 by: -1 do:
            [:index ! aBlock value: (elements at: index)]].
! !

!AnimationElement methodsFor: 'enumerating' stamp: 'm3r ;
withAllSubelementsDo: aBlock
    aBlock value: self.
    self subelementsDo:
        [:element ! element withAllSubelementsDo: aBlock].
! !

!AnimationElement methodsFor: 'enumerating' stamp: 'm3r
withSubelementsDo: aBlock
    aBlock value: self.
    self subelementsDo: aBlock
! !

!AnimationElement methodsFor: 'testing' stamp: 'm3r
containsActive
    self withAllSubelementsDo:
        [:element ! element isActive ifTrue: [^true]].
    ^false! !

!AnimationElement methodsFor: 'testing' stamp: 'm3r .
containsVisible
    self withAllSubelementsDo:
        [:element ! element isVisible ifTrue: [^true]].
    ^false! !

!AnimationElement methodsFor: 'testing' stamp: 'm3r
isActive
    ^self selfActivity ifNil:
        [self isTopContext
            ifFalse: [self context isActive]
            ifTrue: [self selfActivity: self defaultActivity]]! !

!AnimationElement methodsFor: 'testing' stamp: 'm3r 2.
isCompletelyDirty
    ^dirty = bounds! !

!AnimationElement methodsFor: 'testing' stamp: 'm3r
isCounter
    ^false! !

!AnimationElement methodsFor: 'testing' stamp: 'm3r
isDirty
    ^dirty — false! !

!AnimationElement methodsFor: 'testing' stamp: 'm3r
isEnabled
    ^self selfEnable ifNil:
        [self isTopContext
            ifFalse: [self context isEnabled]
            ifTrue: [self selfEnable: self defaultEnable]]! !

!AnimationElement methodsFor: 'testing' stamp: 'm3r .
isHilited
    ^self selfHilite ifNil:
        [self isTopContext
            ifFalse: [self context isHilited]
            ifTrue: [self selfHilite: self defaultHilite]]! !

!AnimationElement methodsFor: 'testing' stamp: 'm3r ;
isTopContext
    ^self context == nil! !

!AnimationElement methodsFor: 'testing' stamp: 'm3r
isVisible
    ^self selfVisibility ifNil:
        [self isTopContext
            ifFalse: [self context isVisible]
            ifTrue: [self selfVisibility: self defaultVisibility]]! !

!AnimationElement methodsFor: 'display profiling' stamp: 'm3r,
depth
    ^self findPropertyAt: #depth ifAbsentAtTopContextPut:
        [:topContext ! self defaultDepth]
! !

!AnimationElement methodsFor: 'display profiling' stamp: 'm3r
depth: depth
```

```
    self depth = depth ifTrue: [^self].
    self
        propertyAt: #depth put: depth
        sharedPropertyChanged: #depth.
! !

!AnimationElement methodsFor: 'display profiling' stamp: 'm3r
ensureDisplayProfile! !

!AnimationElement methodsFor: 'display profiling' stamp: 'm3r
frameCount
    ^self stepCount + 1
! !

!AnimationElement methodsFor: 'display profiling' stamp: 'm3r
frameCount: anInteger
    ^self stepCount: anInteger - 1
! !

!AnimationElement methodsFor: 'display profiling' stamp: 'm3r
hvFillRatio
    ^self findPropertyAt: #hvFillRatio ifAbsentAtTopContextPut:
        [:topContext | self defaultHVfillRatio]
! !

!AnimationElement methodsFor: 'display profiling' stamp: 'm3r
hvFillRatio: aRatio
    self hvFillRatio = aRatio ifTrue: [^self].
    self
        propertyAt: #hvFillRatio put: aRatio;
        sharedPropertyChanged: #hvFillRatio.
! !

!AnimationElement methodsFor: 'display profiling' stamp: 'm3r
hvMinBorderGap
    ^self findPropertyAt: #hvMinBorderGap ifAbsentAtTopContextPut:
        [:topContext | self defaultHVminBorderGap]
! !

!AnimationElement methodsFor: 'display profiling' stamp: 'm3r
hvMinBorderGap: gapPoint
    self hvMinBorderGap = gapPoint ifTrue: [^self].
    self
        propertyAt: #hvMinBorderGap put: gapPoint;
        sharedPropertyChanged: #hvMinBorderGap.
! !

!AnimationElement methodsFor: 'display profiling' stamp: 'm3r
stepCount
    ^self findPropertyAt: #stepCount ifAbsentAtTopContextPut:
        [:topContext | self defaultStepCount]
! !

!AnimationElement methodsFor: 'display profiling' stamp: 'm3r
stepCount: anInteger
    | aStep |
    (aStep _ self stepCount) = anInteger ifTrue: [^self].
    self
        propertyAt: #stepCount put: anInteger;
        sharedPropertyChanged: #stepCount with: aStep.
! !

!AnimationElement methodsFor: 'attributes setting' stamp: 'm3r
beActive
    | previousActivity |
    previousActivity _ self isActive.
    self selfActivity: true.
    previousActivity ifTrue: [^self].
    self sharedAspectChanged: #selfActivity with: true.
! !

!AnimationElement methodsFor: 'attributes setting' stamp: 'm3r
beDisabled
    | previousEnable |
    previousEnable _ self isEnabled.
    self selfEnable: false.
    previousEnable ifTrue: [^self].
    self sharedAspectChanged: #selfEnable with: false.
! !

!AnimationElement methodsFor: 'attributes setting' stamp: 'm3r
beEnabled
    | previousEnable |
    previousEnable _ self isEnabled.
    self selfEnable: true.
    previousEnable ifTrue: [^self].
    self sharedAspectChanged: #selfEnable with: true.
! !

!AnimationElement methodsFor: 'attributes setting' stamp: 'm3r
beHilited
    | previousHilite |
    previousHilite _ self isHilited.
    self selfHilite: true.
    previousHilite ifTrue: [^self].
    self sharedAspectChanged: #selfHilite with: true.
! !

!AnimationElement methodsFor: 'attributes setting' stamp: 'm3r
beInactive
    | previousActivity |
    previousActivity _ self isActive.
    self selfActivity: false.
    previousActivity ifTrue: [^self].
    self sharedAspectChanged: #selfActivity with: false.
! !

!AnimationElement methodsFor: 'attributes setting' stamp: 'm3r
beInvisible
    | previousVisibility |
    previousVisibility _ self isVisible.
    self selfVisibility: false.
    previousVisibility ifTrue: [^self].
    self
        markUncoveredAreaDirtyInContext;
        sharedAspectChanged: #selfVisibility with: false.
! !

!AnimationElement methodsFor: 'attributes setting' stamp: 'm3r
beUnhilited
    | previousHilite |
    previousHilite _ self isHilited.
    self selfHilite: false.
    previousHilite ifTrue: [^self].
    self sharedAspectChanged: #selfHilite with: false.
! !

!AnimationElement methodsFor: 'attributes setting' stamp: 'm3r
beVisible
    | previousVisibility |
    previousVisibility _ self isVisible.
    self selfVisibility: true.
    previousVisibility ifTrue: [^self].
    self sharedAspectChanged: #selfVisibility with: true.
! !
```

```
!AnimationElement methodsFor: 'accessing' stamp: 'm3r
backgroundImage
    ^self findPropertyAt: #backgroundImage ifAbsentAtTopContextPut:
        [:topContext | self defaultBackgroundImage]! !

!AnimationElement methodsFor: 'accessing' stamp: 'm3r
backgroundImage: aColorOrPatternOrForm
    | aBackground |
    (aBackground _ self backgroundImage) = aColorOrPatternOrForm
        ifTrue: [^self].
    self
        propertyAt: #backgroundImage put: aColorOrPatternOrForm;
        sharedPropertyChanged: #backgroundImage with: aBackground.
! !

!AnimationElement methodsFor: 'accessing' stamp: 'm3r
color
    ^self findPropertyAt: #color ifAbsentAtTopContextPut:
        [:topContext | self defaultColor]! !

!AnimationElement methodsFor: 'accessing' stamp: 'm3r
color: aColorPattern
    | aColor |
    (aColor _ self color) = aColorPattern ifTrue: [^self].
    self
        propertyAt: #color put: aColorPattern;
        sharedPropertyChanged: #color with: aColor.
! !

!AnimationElement methodsFor: 'accessing' stamp: 'm3r
creator
    ^self propertyAt: #creator ifAbsent: [self]! !

!AnimationElement methodsFor: 'accessing' stamp: 'm3r
creator: anAnimationObject
    ^self propertyAt: #creator put: anAnimationObject! !

!AnimationElement methodsFor: 'accessing' stamp: 'm3r
id
    ^self propertyAt: #id! !

!AnimationElement methodsFor: 'accessing' stamp: 'm3r
id: anIdentifier
    ^self propertyAt: #id put: anIdentifier! !

!AnimationElement methodsFor: 'accessing' stamp: 'm3r
step
    | context |
    ^self selfStep ifNil:
        [(context _ self context) == nil
            ifTrue: [self selfStep: 0]
            ifFalse: [context step]]! !

!AnimationElement methodsFor: 'accessing' stamp: 'm3r
step: newStep
    | previousStep |
    previousStep _ self step.
    self selfStep: newStep.
    previousStep = newStep ifTrue: [^self].
    self sharedAspectChanged: #step with: previousStep.
! !

!AnimationElement methodsFor: 'private accessing' stamp: 'm3r
selfActivity
    ^isActive! !

!AnimationElement methodsFor: 'private accessing' stamp: 'm3r
selfActivity: aBoolean
    ^isActive _ aBoolean! !

!AnimationElement methodsFor: 'private accessing' stamp: 'm3r
selfEnable
    ^isEnabled! !

!AnimationElement methodsFor: 'private accessing' stamp: 'm3r
selfEnable: aBoolean
    ^isEnabled _ aBoolean! !

!AnimationElement methodsFor: 'private accessing' stamp: 'm3r
selfHilite
    ^isHilited! !

!AnimationElement methodsFor: 'private accessing' stamp: 'm3r
selfHilite: aBoolean
    ^isHilited _ aBoolean! !

!AnimationElement methodsFor: 'private accessing' stamp: 'm3r
selfStep
    ^step! !

!AnimationElement methodsFor: 'private accessing' stamp: 'm3r
selfStep: newStep
    ^step _ newStep! !

!AnimationElement methodsFor: 'private accessing' stamp: 'm3r
selfVisibility
    ^isVisible! !

!AnimationElement methodsFor: 'private accessing' stamp: 'm3r
selfVisibility: aBoolean
    ^isVisible _ aBoolean! !

!AnimationElement methodsFor: 'private accessing' stamp: 'm3r
setContext: aContext
    ^elements isCollection
        ifFalse: [elements _ aContext]
        ifTrue: [elements at: 1 put: aContext]
! !

!AnimationElement methodsFor: 'private accessing' stamp: 'm3r
setGraphics: aGraphicsContext
    ^self propertyAt: #graphics put: aGraphicsContext! !

!AnimationElement methodsFor: 'properties' stamp: 'm3r
findPropertyAt: aSymbol ifAbsent: exceptionAction
    ^self propertyAt: aSymbol ifAbsent:
        [self isTopContext
            ifFalse: [self context
                    findPropertyAt: aSymbol
                    ifAbsent: exceptionAction]
            ifTrue: [exceptionAction value]]! !

!AnimationElement methodsFor: 'properties' stamp: 'm3r
findPropertyAt: aSymbol ifAbsentAtTopContextPut: blockWithTopContextArg
    ^self propertyAt: aSymbol ifAbsent:
        [self isTopContext
            ifFalse: [self context
                    findPropertyAt: aSymbol
                    ifAbsentAtTopContextPut: blockWithTopContextArg]
            ifTrue: [self
                    propertyAt: aSymbol
                    put: (blockWithTopContextArg value: self)]]! !

!AnimationElement methodsFor: 'properties' stamp: 'm3r
findPropertyAt: aSymbol ifAbsentPut: valueBlock
    ^self findPropertyAt: aSymbol ifAbsent:
```

```
    [self propertyAt: aSymbol put: valueBlock value]! !

!AnimationElement methodsFor: 'properties' stamp: 'm3r
propertyAt: aSymbol
    ^self propertyAt: aSymbol ifAbsent: [nil]! !

!AnimationElement methodsFor: 'properties' stamp: 'm3r
propertyAt: aSymbol ifAbsent: exceptionAction
    properties == nil ifTrue: [^exceptionAction value].
    ^properties at: aSymbol ifAbsent: exceptionAction! !

!AnimationElement methodsFor: 'properties' stamp: 'm3r
propertyAt: aSymbol ifAbsentPut: valueAction
    properties == nil ifTrue: [properties _ IdentityDictionary new].
    ^properties at: aSymbol ifAbsentPut: valueAction! !

!AnimationElement methodsFor: 'properties' stamp: 'm3r
propertyAt: aSymbol ifPresent: aOneArgBlock
    ^aOneArgBlock value: (self propertyAt: aSymbol ifAbsent: [^nil])! !

!AnimationElement methodsFor: 'properties' stamp: 'm3r
propertyAt: aSymbol put: value
    properties == nil ifTrue: [properties _ IdentityDictionary new].
    ^properties at: aSymbol put: value! !

!AnimationElement methodsFor: 'displaying' stamp: 'm3r
graphics
    ^self findPropertyAt: #graphics ifAbsent: [nil]! !

!AnimationElement methodsFor: 'displaying' stamp: 'm3r
internalExtent
    | extent width height internalWidth internalHeight hvFillRatio hvMinBorderGap |
    extent _ self extent.
    width _ extent x.
    height _ extent y.
    hvFillRatio _ self hvFillRatio.
    hvMinBorderGap _ self hvMinBorderGap * 2.
    internalWidth _ (width * hvFillRatio x) rounded.
    internalHeight _ (height * hvFillRatio y) rounded.
    internalWidth _ (width - internalWidth) >= hvMinBorderGap x
        ifTrue: [width] ifFalse: [width - hvMinBorderGap x].
    internalHeight _ (height - internalHeight) >= hvMinBorderGap y
        ifTrue: [height] ifFalse: [height - hvMinBorderGap y].
    ^internalWidth @ internalHeight
! !

!AnimationElement methodsFor: 'displaying' stamp: 'm3r
offset
    ^self graphics offset! !

!AnimationElement methodsFor: 'drawing' stamp: 'm3r
drawOn: aGraphics
    "client isVisible ifFalse: [^self]
visible show boarder
active show step number
disabled gray board
hilite invert"
! !

!AnimationElement methodsFor: 'actions'!
actionAt: aSymbol
    ^self actionAt: aSymbol ifAbsent: [nil]
! !

!AnimationElement methodsFor: 'actions'!
actionAt: aSymbol ifAbsent: exceptionAction
    ^self class actions at: aSymbol ifAbsent: exceptionAction
! !

!AnimationElement methodsFor: 'actions'!
actionAt: aSymbol ifPresent: aOneArgBlock
    ^aOneArgBlock value: (self actionsAt: aSymbol ifAbsent: [^nil])
! !

!AnimationElement methodsFor: 'actions' stamp: 'm3r.
actionAt: aSymbol ifPresent: aOneArgBlock ifAbsent: exceptionAction
    ^aOneArgBlock value:
        (self actionAt: aSymbol ifAbsent: [^exceptionAction value])
! !

!AnimationElement methodsFor: 'change notification' stamp: 'm3r
aspect: aSymbol update: aSelector
    (self perform: aSelector) ifTrue:
        [self subelementsDo:
            [:element | element aspect: aSymbol update: aSelector]].! !

!AnimationElement methodsFor: 'change notification' stamp: 'm3r.
aspect: aSymbol update: aSelector with: anObject
    (self perform: aSelector with: anObject) ifTrue:
        [self subelementsDo:
            [:element |
            element aspect: aSymbol update: aSelector with: anObject]].! !

!AnimationElement methodsFor: 'change notification' stamp: 'm3r
aspectChanged: aSymbol
    | doPropagate |
    doPropagate _ self
        actionAt: aSymbol
        ifPresent:
            [:actionData | (actionData value "arg count") = 0 ifFalse:
                [self error: 'Wrong number of args for change action!!"].
            self perform: actionData key]
        ifAbsent: [self defaultChangeAction].
    doPropagate ifFalse: [^self].
    self subelementsDo:
        [:element | element aspectChanged: aSymbol].! !

!AnimationElement methodsFor: 'change notification' stamp: 'm3r
aspectChanged: aSymbol with: anObject
    | doPropagate |
    doPropagate _ self
        actionAt: aSymbol
        ifPresent:
            [:actionData | (actionData value "arg count") caseOf:
                {[0] -> [self perform: (actionData key)].
                 [1] -> [self perform: (actionData key) with: anObject].
                 [2] -> [self perform: (actionData key) with: aSymbol with: anObject]}]
        ifAbsent: [self defaultChangeAction].
    doPropagate ifFalse: [^self].
    self subelementsDo:
        [:element | element aspectChanged: aSymbol with: anObject].! !

!AnimationElement methodsFor: 'change notification' stamp: 'm3r
sharedAspect: aSymbol update: aSelector
    (self perform: aSelector) ifTrue:
        [self subelementsDo:
            [:element | (element perform: aSymbol) == nil ifTrue:
                [element sharedAspect: aSymbol update: aSelector]]].! !

!AnimationElement methodsFor: 'change notification' stamp: 'm3r
sharedAspect: aSymbol update: aSelector with: anObject
    (self perform: aSelector with: anObject) ifTrue:
        [self subelementsDo:
            [:element | (element perform: aSymbol) == nil ifTrue:
                [element sharedAspect: aSymbol update: aSelector with: anObject]]].! !
```

```
!AnimationElement methodsFor: 'change notification' stamp: 'm3r
sharedAspectChanged: aSymbol
    | doPropagate |
    doPropagate _ self
       actionAt: aSymbol
       ifPresent:
          [:actionData | (actionData value "arg count") = 0 ifFalse:
             [self error: 'Wrong number of args for change action!!'].
          self perform: actionData key]
       ifAbsent: [self defaultChangeAction].
    doPropagate ifFalse: [^self].
    self subelementsDo:
       [:element | (element perform: aSymbol) == nil ifTrue:
          [element sharedAspectChanged: aSymbol]].! !

!AnimationElement methodsFor: 'change notification' stamp: 'm3r
sharedAspectChanged: aSymbol with: anObject
    | doPropagate |
    doPropagate _ self
       actionAt: aSymbol
       ifPresent:
          [:actionData | (actionData value "arg count") caseOf:
             {[0] -> [self perform: (actionData key)].
              [1] -> [self perform: (actionData key) with: anObject].
              [2] -> [self perform: (actionData key) with: aSymbol with: anObject]}]
       ifAbsent: [self defaultChangeAction].
    doPropagate ifFalse: [^self].
    self subelementsDo:
       [:element | (element perform: aSymbol) == nil ifTrue:
          [element sharedAspectChanged: aSymbol with: anObject]].! !

!AnimationElement methodsFor: 'change notification' stamp: 'm3r
sharedProperty: aSymbol update: aSelector
    (self perform: aSelector) ifTrue:
       [self subelementsDo:
          [:element | (element propertyAt: aSymbol) == nil ifTrue:
             [element sharedProperty: aSymbol update: aSelector]]].! !

!AnimationElement methodsFor: 'change notification' stamp: 'm3r
sharedProperty: aSymbol update: aSelector with: anObject
    (self perform: aSelector with: anObject) ifTrue:
       [self subelementsDo:
          [:element | (element propertyAt: aSymbol) == nil ifTrue:
             [element sharedProperty: aSymbol update: aSelector with: anObject]]].! !

!AnimationElement methodsFor: 'change notification' stamp: 'm3r
sharedPropertyChanged: aSymbol
    | doPropagate |
    doPropagate _ self
       actionAt: aSymbol
       ifPresent:
          [:actionData | (actionData value "arg count") = 0 ifFalse:
             [self error: 'Wrong number of args for change action!!'].
          self perform: actionData key]
       ifAbsent: [self defaultChangeAction].
    doPropagate ifFalse: [^self].
    self subelementsDo:
       [:element | (element propertyAt: aSymbol) == nil ifTrue:
          [element sharedPropertyChanged: aSymbol]].! !

!AnimationElement methodsFor: 'change notification' stamp: 'm3r
sharedPropertyChanged: aSymbol with: anObject
    | doPropagate |
    doPropagate _ self
       actionAt: aSymbol
       ifPresent:
          [:actionData | (actionData value "arg count") caseOf:
             {[0] -> [self perform: (actionData key)].
              [1] -> [self perform: (actionData key) with: anObject].
              [2] -> [self perform: (actionData key) with: aSymbol with: anObject]}]
       ifAbsent: [self defaultChangeAction].
    doPropagate ifFalse: [^self].
    self subelementsDo:
       [:element | (element propertyAt: aSymbol) == nil ifTrue:
          [element sharedPropertyChanged: aSymbol with: anObject]].! !

!AnimationElement methodsFor: 'handling change' stamp: 'm3r
activityChangedTo: aBoolean
    self notYetImplemented.

    "NOTE:  This method is currently not used.
     I haven't finalized what activity means.
     Inactive means either it doesn't currently respond to
     animation activity or it doesn't respond to user input activity."
    "^(isActive and: [client isAnimating])
       ifTrue: [self aspectChangedFor: client]
       ifFalse: [nil]"
          "The client will only be dirty if it has changed in appearance."
    ^true
! !

!AnimationElement methodsFor: 'handling change' stamp: 'm3r
aspectChanged
    self markVisibleAreaDirty.
    ^true
! !

!AnimationElement methodsFor: 'handling change' stamp: 'm3r
basicAspectChanged
    self markDirty: self maxVisibleArea.
    ^true
! !

!AnimationElement methodsFor: 'handling change' stamp: 'm3r
boundsChanged
    ^self profileAspectChanged! !

!AnimationElement methodsFor: 'handling change' stamp: 'm3r
coordinateSystemChangedBy: deltaPoint
    self markVisibleAreaDirty.
    ^false
! !

!AnimationElement methodsFor: 'handling change' stamp: 'm3r
defaultChangeAction
    ^true! !

!AnimationElement methodsFor: 'handling change'!
hiliteChanged
    ^self basicAspectChanged
! !

!AnimationElement methodsFor: 'handling change' stamp: 'm3r
profileAspectChanged
    ^self
       ensureDisplayProfile;
       aspectChanged
! !

!AnimationElement methodsFor: 'handling change' stamp: 'm3r
spatialAspectChanged
    self markVisibleAreaDirty.
    ^false
! !

!AnimationElement methodsFor: 'handling change'!
```

```
visibilityChangedTo: aBoolean
    aBoolean ifFalse: [^true].
    ^self aspectChanged
! !

!AnimationElement methodsFor: 'area calculation' stamp: 'm3r
coveredArea
    ^self merge: self dirtyArea into: self visibleArea! !

!AnimationElement methodsFor: 'area calculation' stamp: 'm3r
dirtyArea
    ^dirty isBoolean ifTrue: [nil] ifFalse: [dirty copy]
! !

!AnimationElement methodsFor: 'area calculation' stamp: 'm3r
drawingArea
    ^nil! !

!AnimationElement methodsFor: 'area calculation' stamp: 'm3r .
maxDrawingArea
    ^self basicDrawingArea
! !

!AnimationElement methodsFor: 'area calculation' stamp: 'm3r
maxVisibleArea
    ^self basicVisibleArea
! !

!AnimationElement methodsFor: 'area calculation' stamp: 'm3r
validArea
    ^self areaWithinContext
! !

!AnimationElement methodsFor: 'area calculation' stamp: 'm3r
visibleArea
    self isVisible ifFalse: [^nil].
    ^self clip: self drawingArea by: self validArea! !

!AnimationElement methodsFor: 'private area calculation' stamp: 'm3r
areaWithinContext
    | context |
    (context _ self context) == nil ifTrue: [^self bounds].
    ^self clip: (self convertAreaFromContext: context bounds) by: bounds
! !

!AnimationElement methodsFor: 'private area calculation' stamp: 'm3r
basicDrawingArea
    ^self bounds
! !

!AnimationElement methodsFor: 'private area calculation' stamp: 'm3r
basicVisibleArea
    self isVisible ifFalse: [^nil].
    ^self clip: self basicDrawingArea by: self validArea! !

!AnimationElement methodsFor: 'private area calculation' stamp: 'm3r
mergeCoveredAreaWith: baseArea
    ^self
        mergeCoveredAreaWith: baseArea
        clip: self infiniteBounds
        with: self isVisible ! !

!AnimationElement methodsFor: 'private area calculation' stamp: 'm3r
mergeCoveredAreaWith: baseArea clip: clipArea with: sharedVisibility
    | clippedValidArea visibility rawCoveredArea selfCoveredArea
      coveredArea delta elementLocation elementArea |
    (clippedValidArea _ self clip: self validArea by: clipArea) == nil ifTrue: [^nil].
        "If true, I and my subelements are completely out of view."

    (self does: baseArea contain: clippedValidArea) ifTrue: [^baseArea].
        "If true, neither I or my subelements can increase the baseArea."

    (visibility _ self selfVisibility) == nil ifTrue: [visibility _ sharedVisibility].

    visibility ifTrue:
        [rawCoveredArea _ self merge: self drawingArea into: self dirtyArea.

        (self does: rawCoveredArea contain: clippedValidArea) ifTrue:
            [^self merge: clippedValidArea into: baseArea].
                "rawCoveredArea has already filled up the clippedValidArea,
                 so there is no reason to enumerate any subelements."

        selfCoveredArea _ self clip: rawCoveredArea by: clippedValidArea.
        coveredArea _ self merge: selfCoveredArea into: baseArea].

    delta _ 000.
    self subelementsDo:
        [:element | elementLocation _ element location.
        coveredArea moveSelfBy: (delta _ elementLocation - delta).
        clippedValidArea moveSelfBy: delta.
            "Move the coveredArea and clippedValidArea into
             the element's coordinate system."

        elementArea _ element
            mergeCoveredAreaWith: coveredArea
            clip: clippedValidArea
            with: visibility.
        coveredArea _ self merge: elementArea into: coveredArea].

    ^coveredArea == nil
        ifTrue: [nil]
        ifFalse: [coveredArea moveSelfBy: elementLocation negated]
                "Restore coveredArea to coordinate system before answering."! !

!AnimationElement methodsFor: 'marking dirty' stamp: 'm3r
markAreaDirty: anArea
    | context |
    anArea == nil ifTrue: [^self].
    (context _ self context) == nil ifFalse: [context markDirty].
    dirty isBoolean
        ifTrue: [dirty _ anArea copy]
        ifFalse: [dirty mergeIntoSelf: anArea clippedBy: bounds]! !

!AnimationElement methodsFor: 'marking dirty' stamp: 'm3r
markBoundsDirty
    self markAreaDirty: bounds! !

!AnimationElement methodsFor: 'marking dirty' stamp: 'm3r
markClean
    dirty _ false.
! !

!AnimationElement methodsFor: 'marking dirty' stamp: 'm3r
markDirty
    | context |
    dirty == false ifTrue:
        [dirty _ true.
        (context _ self context) == nil ifFalse: [context markDirty]].! !

!AnimationElement methodsFor: 'marking dirty' stamp: 'm3r
markUncoveredAreaDirtyInContext
    | context dirtyArea |
    (context _ self context) == nil ifTrue: [^nil].
    dirtyArea _ context dirtyArea.
    dirtyArea _ self convertAreaFromContext: dirtyArea.
    dirtyArea _ self mergeCoveredAreaWith: dirtyArea.
```

```
    ^context markAreaDirty: (self convertAreaToContext: dirtyArea)! !

!AnimationElement methodsFor: 'marking dirty' stamp: 'n3r
markVisibleAreaDirty
    ^self markAreaDirty: self visibleArea! !

!AnimationElement methodsFor: 'marking all dirty' stamp: 'n3r
markAllBoundsDirty
    self withAllSubelementsDo: [:element | element markBoundsDirty].
! !

!AnimationElement methodsFor: 'marking all dirty' stamp: 'n3r
markAllClean
    self withAllSubelementsDo: [:element | element markClean].
! !

!AnimationElement methodsFor: 'marking all dirty' stamp: 'n3r
markAllDirty
    self withAllSubelementsDo: [:element | element markDirty].
! !

!AnimationElement methodsFor: 'absolute spatial accessing' stamp: 'n3r
absoluteBounds
    ^self bounds moveSelfBy: self absoluteLocation! !

!AnimationElement methodsFor: 'absolute spatial accessing' stamp: 'n3r
absoluteLocation
    ^self absoluteOrigin! !

!AnimationElement methodsFor: 'absolute spatial accessing' stamp: 'n3r
absoluteOrigin
    ^self convertPointToAbsolute: 0@0! !

!AnimationElement methodsFor: 'absolute spatial accessing' stamp: 'n3r
convertPointToAbsolute: aPoint
    | answer currentContext |
    answer _ aPoint.
    currentContext _ self.
    [answer _ answer + currentContext location.
    currentContext isTopContext]
        whileFalse: [currentContext _ currentContext context].
    ^answer! !

!AnimationElement methodsFor: 'external spatial accessing' stamp: 'n3r
convertAreaFromContext: anArea
    ^anArea moveSelfBy: self location negated! !

!AnimationElement methodsFor: 'external spatial accessing' stamp: 'n3r
convertAreaToContext: anArea
    ^anArea moveSelfBy: self location! !

!AnimationElement methodsFor: 'external spatial accessing' stamp: 'n3r
location
    "Answers my distance from my context's origin."

    ^location! !

!AnimationElement methodsFor: 'external spatial accessing' stamp: 'n3r
location: aPoint
    "Sets my distance from my context's origin."

    "My context may have changed by having new areas that are now covered or
    uncovered by me."

    | previousLocation |
    (previousLocation _ self location) = aPoint ifFalse:
        [self
            markUncoveredAreaDirtyInContext;
            setLocation: aPoint;
            aspectChanged: #location with: previousLocation].
! !

!AnimationElement methodsFor: 'external spatial accessing' stamp: 'n3r
locationBounds
    "Answers my bounds in the coordinate system of my context."

    "#bounds returns a new rect so no need to make a new one."
    ^self bounds moveSelfBy: self location! !

!AnimationElement methodsFor: 'external spatial accessing' stamp: 'n3r
locationBounds: aRect
    "Sets my bounds in the coordinate system of my context.  My origin offset
    from the top left of my bounds stays the same.  This method is normally
    called before the origin offset has been set."

    "My image may have changed by growing, shrinking, or otherwise changing in
    shape.  Furthermore my context may have changed by having new areas that
    are now covered or uncovered by me."

    | previousLocationBounds |
    (previousLocationBounds _ self locationBounds) = aRect ifFalse:
        [self
            markUncoveredAreaDirtyInContext;
            setLocationBounds: aRect;
            aspectChanged: #locationBounds with: previousLocationBounds].
! !

!AnimationElement methodsFor: 'external spatial accessing' stamp: 'n3r
moveLocationBy: delta
    "Moves my distance from my context's origin by delta."

    self location: (self location + delta).! !

!AnimationElement methodsFor: 'internal spatial accessing' stamp: 'n3r
bounds
    ^bounds copy! !

!AnimationElement methodsFor: 'internal spatial accessing' stamp: 'n3r
bounds: aRect
    "My context may have changed by having new areas that are now covered or
    uncovered by me."

    "My image may have changed by growing, shrinking, or otherwise changing in
    shape.  Furthermore my context may have changed by having new areas that
    are now covered or uncovered by me."

    | previousBounds |
    (previousBounds _ bounds) = aRect ifFalse:
        [self
            markUncoveredAreaDirtyInContext;
            setBounds: aRect;
            aspectChanged: #bounds with: previousBounds].
! !

!AnimationElement methodsFor: 'internal spatial accessing' stamp: 'n3r
boundsOffset
    "Answers the offset of the top left of my bounds from my origin."

    ^bounds origin! !

!AnimationElement methodsFor: 'internal spatial accessing' stamp: 'n3r
extent
    ^bounds extent! !

!AnimationElement methodsFor: 'internal spatial accessing' stamp: 'n3r
moveCoordinateSystemBy: delta
```

```
    "Move the offset of my local origin by delta.  My subelements locations are
     adjusted so that they remain aligned to my old origin."

    "My image may have changed because my drawing is shifted."

    delta = 0@0 ifFalse:
        [self
            setOriginOffsetBy: delta;
            subelementsDo:
                [:element | element setLocation: element location - delta];
            aspectChanged: #coordinateSystem with: delta].
! !

!AnimationElement methodsFor: 'internal spatial accessing' stamp: 'm3r
moveCoordinateSystemTo: aPoint
    "Sets the offset of my local origin from the top left (origin) of my bounds.
     My subelements locations are adjusted so that they remain aligned to my
     old origin."

    self moveCoordinateSystemBy: (aPoint + self boundsOffset).! !

!AnimationElement methodsFor: 'internal spatial accessing' stamp: 'm3r
moveOriginBy: delta
    "Move the offset of my local origin by delta.  My subelements are shifted to
     remain at their same locations from my origin."

    "My image may have changed because my drawing is shifted as well as the drawing
     of my subelements."

    delta = 0@0 ifFalse:
        [self subelementsDo: [:element | element markUncoveredAreaDirtyIn: context].
        self
            setOriginOffsetBy: delta;
            aspectChanged: #origin with: delta].
! !

!AnimationElement methodsFor: 'internal spatial accessing' stamp: 'm3r
moveOriginTo: aPoint
    "Sets the offset of my local origin from the top left (origin) of my bounds.
     All of my subelements are shifted to remain at their same locations from my
     origin."

    self moveOriginBy: (aPoint + self boundsOffset).! !

!AnimationElement methodsFor: 'internal spatial accessing' stamp: 'm3r
originOffset
    "Answers the offset of my origin from the top left of my bounds."

    ^self boundsOffset negated! !

!AnimationElement methodsFor: 'private spatial methods' stamp: 'm3r
getBounds
    ^bounds! !

!AnimationElement methodsFor: 'private spatial methods' stamp: 'm3r
setBounds: aRect
    bounds _ aRect.! !

!AnimationElement methodsFor: 'private spatial methods' stamp: 'm3r
setLocation: aPoint
    ^location _ aPoint.! !

!AnimationElement methodsFor: 'private spatial methods' stamp: 'm3r
setLocationBounds: aRect
    location _ aRect origin - self boundsOffset.
    bounds setExtent: aRect extent.! !

!AnimationElement methodsFor: 'private spatial methods' stamp: 'm3r
setOriginOffsetBy: delta
    bounds moveSelfBy: delta negated.
    location _ location + delta.! !

!AnimationElement methodsFor: 'private' stamp: 'm3r
initialize
    super initialize.
    bounds _ 0@0 extent: 0@0.
    location _ 0@0.
    dirty _ false.
! !

!AnimationElement methodsFor: 'private' stamp: 'm3r
setExtent: anExtent depth: aDepth
    self
        setGraphics: (GraphicsContext extent: anExtent depth: aDepth);
        setBounds: (0@0 corner: anExtent).! !

!AnimationElement methodsFor: 'private' stamp: 'm3r
setGraphics: aDisplayMedium bounds: aRect
    | graphics |
    graphics _ aDisplayMedium asGraphicsContext.
    self
        setGraphics: graphics;
        setBounds: (graphics clipBoundsWith: aRect).! !

!AnimationElement methodsFor: 'defaults' stamp: 'm3r
defaultActivity
    ^false! !

!AnimationElement methodsFor: 'defaults' stamp: 'm3r
defaultBackgroundImage
    ^Color lightGray! !

!AnimationElement methodsFor: 'defaults' stamp: 'm3r
defaultColor
    ^Color black! !

!AnimationElement methodsFor: 'defaults' stamp: 'm3r
defaultDepth
    ^4
! !

!AnimationElement methodsFor: 'defaults' stamp: 'm3r
defaultEnable
    ^false! !

!AnimationElement methodsFor: 'defaults' stamp: 'm3r
defaultHiliteBorderGap
    ^1@1
! !

!AnimationElement methodsFor: 'defaults' stamp: 'm3r
defaultHiliteRatio
    ^1@1
! !

!AnimationElement methodsFor: 'defaults' stamp: 'm3r
defaultMainBorderGap
    ^1@1
! !

!AnimationElement methodsFor: 'defaults' stamp: 'm3r
defaultMlite
    ^false! !

!AnimationElement methodsFor: 'defaults' stamp: 'm3r
defaultStepCount
```

```
    ^15
! !

!AnimationElement methodsFor: 'defaults' stamp: 'n3r
defaultVisibility
    ^false! !


!AnimationObject class methodsFor: 'instance creation' stamp: 'n3r
new
    ^super new initialize! !

!AnimationObject class methodsFor: 'accessing' stamp: 'n3r
infiniteBounds
    ^InfiniteBounds
! !

!AnimationObject class methodsFor: 'color mapping' stamp: 'n3r
calcColorMapFrom: colors forDepth: dstDepth
    | count mapSize colorMap baseColor |
    count _ colors size - 1.
    colorMap _ Bitmap new: (mapSize _ count * 2).
    dstDepth = 32
        ifTrue: [baseColor _ colors at: colors size.
                colorMap at: 1 put: 0.
                2 to: mapSize do:
                    [:level | colorMap at: level put:
                        ((baseColor alpha: (level / mapSize))
                            pixelWordForDepth: 32)]]
        ifFalse: [colors inject: count into:
                    [:index :color | colorMap
                        at: index put: (color pixelValueForDepth: dstDepth).
                    index + 1]].
    ^colorMap! !

!AnimationObject class methodsFor: 'color mapping' stamp: 'n3r
calcColorMapsFor: color at: srcDepth
    | count colors |
    count _ 1 bitShift: srcDepth.
    colors _ color lightShades: count.
    colors _ colors copyFrom: (count / 2) rounded to: count.
    #(1 2 32 4 16 8) collect:
        [:dstDepth | self calcColorMapFrom: colors forDepth: dstDepth]! !

!AnimationObject class methodsFor: 'color mapping' stamp: 'n3r
colorMapsFor: color at: srcDepth
    ColorMapsDictionary ifNil: [ColorMapsDictionary _ Dictionary new].
    ^ColorMapsDictionary at: (Point x: srcDepth y: color) ifAbsentPut:
        [self calcColorMapsFor: color at: srcDepth]! !

!AnimationObject class methodsFor: 'class initialization' stamp: 'n3r
ensure
    InfiniteBounds == nil ifTrue:
        [self initialize].! !

!AnimationObject class methodsFor: 'class initialization' stamp: 'n3r
initialize
        "AnimationObject initialize."

    Smalltalk at: #null ifAbsentPut: [#NonObjectPlaceHolder].
    InfiniteBounds _ Rectangle
        origin: SmallInteger minVal asPoint
        corner: SmallInteger maxVal asPoint.
! !


!AnimationElement class reorganize!
('instance creation' extent:depth: in: on: on:inBounds:)
('accessing' actions profileClass)
('actions' addAction:for: addStandardActionFor: removeActionAt:ifAbsent: removeActions standardChange:
('class initialization' ensure initializeActions rebuildAllActions)
!


!AnimationElement class methodsFor: 'instance creation' stamp: 'n3r
extent: extent depth: depth
    ^self new setExtent: extent depth: depth
! !

!AnimationElement class methodsFor: 'instance creation' stamp: 'n3r
in: aContext
    ^aContext == nil
        ifTrue: [self new]
        ifFalse: [aContext addSubelement: self new]
! !

!AnimationElement class methodsFor: 'instance creation' stamp: 'n3r
on: aDisplayMedium
    ^self setGraphics: aDisplayMedium bounds: aDisplayMedium boundingBox
! !

!AnimationElement class methodsFor: 'instance creation' stamp: 'n3r
on: aDisplayMedium inBounds: aRect
    ^self new setGraphics: aDisplayMedium bounds: aRect
! !

!AnimationElement class methodsFor: 'accessing' stamp: 'n3r
actions
    ^actions! !

!AnimationElement class methodsFor: 'accessing' stamp: 'n3r
profileClass
    self subclassResponsibility.! !

!AnimationElement class methodsFor: 'actions' stamp: 'n3r'
addAction: anAction for: anAspect
    ^self actions at: anAspect put: (anAction -> anAction numArgs)
! !

!AnimationElement class methodsFor: 'actions' stamp: 'n3r
addStandardActionFor: anAspect
    ^self actions at: anAspect put: (self standardChangeSelector -> 0)
! !

!AnimationElement class methodsFor: 'actions' stamp: 'n3r
removeActionAt: anAspect ifAbsent: exceptionBlock
    ^self actions remove: anAspect ifAbsent: exceptionBlock
! !

!AnimationElement class methodsFor: 'actions' stamp: 'n3r
removeActions
    actions _ nil.! !

!AnimationElement class methodsFor: 'actions' stamp: 'n3r
standardChangeSelector
    ^#aspectChanged
! !

!AnimationElement class methodsFor: 'class initialization' stamp: 'n3r
ensure
    self actions == nil ifTrue:
        [super ensure.
        self initializeActions].! !

!AnimationElement class methodsFor: 'class initialization' stamp: 'n3r
initializeActions
```

```
        "AnimationElement rebuildAllActions."
    actions _ IdentityDictionary new.
    ^self
        addStandardActionFor: #color;
        addStandardActionFor: #backgroundImage;
        addStandardActionFor: #selfEnable;
        addAction: #activityChangedTo:        for: #selfActivity;
        addAction: #visibilityChangedTo:        for: #selfVisibility;
        addAction: #hiliteChanged             for: #selfHilite;

        addAction: #profileAspectChanged      for: #depth;
        addAction: #profileAspectChanged      for: #stepCount;
        addAction: #boundsChanged             for: #hvFillRatio;
        addAction: #boundsChanged             for: #hvMinBorderGap;

        addAction: #boundsChanged             for: #bounds;
        addAction: #boundsChanged             for: #locationBounds;
        addAction: #spatialAspectChanged      for: #location;
        addAction: #spatialAspectChanged      for: #origin;
        addAction: #coordinateSystemChangedBy:   for: #coordinateSystem;
        actions
! !

!AnimationElement class methodsFor: 'class initialization' stamp: 'n3r
rebuildAllActions
    AnimationElement withAllSubclassesDo:
        [:class | class initializeActions]! !


!Boolean methodsFor: 'testing' stamp: 'n3r
isBoolean
    ^true! !


!CounterAE reorganize!
('accessing' digitCount mapIntoRangeAction maxValue maxValue: minValue minValue: range value value: va
('testing' doAnimateValueChange: isCounter passesOverflow)
('incrementing' decrement increment incrementBy:ifFail:)
('range mapping' check:ifNotInRange: mapIntoRange:)
('range mapping actions' errorIfOutOfRange:ValueOverflowIncrement: noChangeIfOutOfRange:ValueOverflow
('value changed actions' animateAlways: animateIfDifferent: animateNever:)
('handling change' valueChangedFrom:)
('defaults' defaultMapIntoRangeAction defaultValueChangedAction)
('private' initialize setMagnitude: setMaxValue: setMinValue: setValue:)
('enumerating' subcountersLowToHighDo:)
('FD display profiling' defaultSegmentLWratio defaultTipLWratio segmentLWratio segmentLWratio: tipLWra
('utility functions' digitCountOf:)


!CounterAE methodsFor: 'accessing' stamp: 'n3r
digitCount
    ^self digitCountOf: self maxValue! !

!CounterAE methodsFor: 'accessing' stamp: 'n3r
mapIntoRangeAction
    ^self findPropertyAt: #mapIntoRangeAction ifAbsentAtTopContextPut:
        [:topContext | self defaultMapIntoRangeAction]! !

!CounterAE methodsFor: 'accessing' stamp: 'n3r
maxValue
    ^self propertyAt: #maxValue ifAbsent: [0]! !

!CounterAE methodsFor: 'accessing' stamp: 'n3r
maxValue: max
    self setMaxValue: max.
    self value > max ifTrue: [self value: max].! !

!CounterAE methodsFor: 'accessing' stamp: 'n3r
minValue
    ^self propertyAt: #minValue ifAbsent: [0]! !

!CounterAE methodsFor: 'accessing' stamp: 'n3r
minValue: min
    self setMinValue: min.
    self value < min ifTrue: [self value: min].! !

!CounterAE methodsFor: 'accessing' stamp: 'n3r
range
    ^self maxValue - self minValue + 1! !

!CounterAE methodsFor: 'accessing' stamp: 'n3r
value
    ^magnitude! !

!CounterAE methodsFor: 'accessing' stamp: 'n3r
value: newValue
    ^self value: newValue ifFail: [nil]! !

!CounterAE methodsFor: 'accessing' stamp: 'n3r
value: aValue ifFail: exceptionAction
    self incrementBy: (aValue - self value) ifFail: exceptionAction.! !

!CounterAE methodsFor: 'accessing' stamp: 'n3r
valueChangedAction
    ^self findPropertyAt: #valueChangedAction ifAbsentAtTopContextPut:
        [:topContext | self defaultValueChangedAction]! !

!CounterAE methodsFor: 'testing' stamp: 'n3r
doAnimateValueChange: previousValue
    ^self valueChangedAction value: self value: previousValue! !

!CounterAE methodsFor: 'testing' stamp: 'n3r
isCounter
    ^true! !

!CounterAE methodsFor: 'testing' stamp: 'n3r
passesOverflow
    ^self mapIntoRangeAction == #wrapWithOverflow! !

!CounterAE methodsFor: 'incrementing' stamp: 'n3r
decrement
    self incrementBy: -1 ifFail: [].! !

!CounterAE methodsFor: 'incrementing' stamp: 'n3r
increment
    self incrementBy: 1 ifFail: [].! !

!CounterAE methodsFor: 'incrementing' stamp: 'n3r
incrementBy: delta ifFail: exceptionAction
    | context value mapAction mappingSucceeded adjustedDelta overflow newValue |
    value _ self value.
    mapAction _ self mapIntoRangeAction.
    mappingSucceeded _ mapAction value: self value: (delta + value) value:
        [:aNewValue :anOverflow :anIncrement |
        newValue _ aNewValue.
        overflow _ anOverflow.
        adjustedDelta _ anIncrement].
    mappingSucceeded ifFalse: [^exceptionAction value].
    ((context _ self context) == nil and: [context isCounter])
        ifTrue: [context incrementBy: adjustedDelta ifFail: exceptionAction]
        ifFalse: [self setValue: newValue.
                ^overflow].
    ^overflow
! !
```

```
!CounterAE methodsFor: 'range mapping' stamp: 'm3r
check: aValue ifNotInRange: overflowBlock
    | min max |
    ^true caseOf:
        {[aValue < (min _ self minValue)] -> [overflowBlock value: aValue - m[n].
        [aValue > (max _ self maxValue)] -> [overflowBlock value: aValue - max]}
        otherwise: [aValue]! !

!CounterAE methodsFor: 'range mapping' stamp: 'm3r
mapIntoRange: aValue
    ^self mapIntoRangeAction value: self value: aValue! !

!CounterAE methodsFor: 'range mapping actions' stamp: 'm3r
errorIfOutOfRange: aValue ValueOverflowIncrement: threeArgAnswerBlock
    self check: aValue ifNotInRange: [^false].
    threeArgAnswerBlock value: aValue value: 0 value: (aValue - self value).
    ^true ! !

!CounterAE methodsFor: 'range mapping actions' stamp: 'm3r
noChangeIfOutOfRange: aValue ValueOverflowIncrement: threeArgAnswerBlock
    | newValue |
    newValue _ self check: aValue ifNotInRange: [self value].
    threeArgAnswerBlock
        value: newValue value: 0 value: (newValue - self value).
    ^true ! !

!CounterAE methodsFor: 'range mapping actions' stamp: 'm3r
pinAtRange: aValue ValueOverflowIncrement: threeArgAnswerBlock
    | newValue |
    newValue _ self check: aValue ifNotInRange:
        [:overflow | overflow > 0
            ifTrue: [self maxValue] ifFalse: [self minValue]].
    threeArgAnswerBlock
        value: newValue value: 0 value: (newValue - self value).
    ^true ! !

!CounterAE methodsFor: 'range mapping actions' stamp: 'm3r
wrapWithOverflow: aValue ValueOverflowIncrement: threeArgAnswerBlock
    | overflow newValue |
    overflow _ 0.
    newValue _ self check: aValue ifNotInRange:
        [:anOverflow | (overflow _ anOverflow) > 0
            ifTrue: [self minValue + overflow - 1]
            ifFalse: [self maxValue - overflow + 1]].
    threeArgAnswerBlock
        value: newValue value: overflow value: (aValue - self value).
    ^true! !

!CounterAE methodsFor: 'range mapping actions' stamp: 'm3r
wrapWithoutOverflow: aValue ValueOverflowIncrement: threeArgAnswerBlock
    | newValue |
    newValue _ self check: aValue ifNotInRange:
        [:overflow | overflow > 0
            ifTrue: [self minValue + overflow - 1]
            ifFalse: [self maxValue - overflow + 1]].
    threeArgAnswerBlock
        value: newValue value: 0 value: (newValue - self value).
    ^true ! !

!CounterAE methodsFor: 'value changed actions' stamp: 'm3r
animateAlways: previousValue
    ^true! !

!CounterAE methodsFor: 'value changed actions' stamp: 'm3r
animateIfDifferent: previousValue
    ^self value ~= previousValue! !

!CounterAE methodsFor: 'value changed actions' stamp: 'm3r
animateNever: previousValue
    ^false! !

!CounterAE methodsFor: 'handling change' stamp: 'm3r'
valueChangedFrom: previousValue
    | remaining range |
    (self doAnimateValueChange: previousValue) ifFalse: [^false].
    self aspectChanged.

    remaining _ self value.
    self subelementsDo:
        [:element | element isCounter
            ifFalse: [element aspectChanged: #value with: previousValue]
            ifTrue: [range _ element range.
                element setValue: remaining \\ range.
                remaining _ remaining // range]].
    ^false! !

!CounterAE methodsFor: 'defaults' stamp: 'm3r
defaultMapIntoRangeAction
    ^#wrapWithOverflow:ValueOverflowIncrement:! !

!CounterAE methodsFor: 'defaults' stamp: 'm3r.
defaultValueChangedAction
    ^#animateAlways:
! !

!CounterAE methodsFor: 'private' stamp: 'm3r'
initialize
    super initialize.
    magnitude _ 0.! !

!CounterAE methodsFor: 'private' stamp: 'm3r
setMagnitude: aMagnitude
    magnitude _ aMagnitude.
! !

!CounterAE methodsFor: 'private' stamp: 'm3r
setMaxValue: max
    self propertyAt: #maxValue put: max! !

!CounterAE methodsFor: 'private' stamp: 'm3r'
setMinValue: min
    self propertyAt: #minValue put: min! !

!CounterAE methodsFor: 'private' stamp: 'm3r
setValue: newValue
    | previousValue |
    previousValue _ self value.
    self setMagnitude: newValue.
    self valueChangedFrom: previousValue.
! !

!CounterAE methodsFor: 'enumerating' stamp: 'm3r
subcountersLowToHighDo: aBlock
    self subelementsDo:
        [:element | element isCounter ifTrue: [aBlock value: element]].
! !

!CounterAE methodsFor: 'FD display profiling' stamp: 'm3r
defaultSegmentLWratio
    ^4.0! !

!CounterAE methodsFor: 'FD display profiling' stamp: 'm3r
defaultTipLNratio
    ^1.0! !

!CounterAE methodsFor: 'FD display profiling' stamp: 'm3r.
```

```
segmentLWratio
    ^self findPropertyAt: #segmentLWratio ifAbsentAtTopContextPut:
        [:topContext | self defaultSegmentLWratio]
! !

!CounterAE methodsFor: 'FD display profiling' stamp: 'm3r
segmentLWratio: aRatio
    self segmentLWratio = aRatio ifTrue: [^self].
    self
        propertyAt: #segmentLWratio put: aRatio;
        sharedPropertyChanged: #segmentLWratio.
! !

!CounterAE methodsFor: 'FD display profiling' stamp: 'm3r
tipLWratio
    ^self findPropertyAt: #tipLWratio ifAbsentAtTopContextPut:
        [:topContext | self defaultTipLWratio]
! !

!CounterAE methodsFor: 'FD display profiling' stamp: 'm3r
tipLWratio: aRatio
    self tipLWratio = aRatio ifTrue: [^self].
    self
        propertyAt: #tipLWratio put: aRatio;
        sharedPropertyChanged: #tipLWratio.
! !

!CounterAE methodsFor: 'utility functions' stamp: 'm3r
digitCountOf: anInteger
    ^anInteger = 0
        ifTrue: [1]
        ifFalse: [((anInteger log: 10) + 1) truncated]
! !


!CounterAE class methodsFor: 'class initialization' stamp: 'm3r
initializeActions
        "AnimationElement rebuildAllActions."
    super initializeActions.
    ^self
        addAction: #valueChangedFrom: for: #value;
        actions! !


!DigitAE methodsFor: 'accessing' stamp: 'm3r
maxValue
    ^9! !

!DigitAE methodsFor: 'accessing' stamp: 'm3r
minValue
    ^0! !

!DigitAE methodsFor: 'private' stamp: 'm3r
setMaxValue: max
    self error: 'maxValue for a DigitAE is always 9!!'! !

!DigitAE methodsFor: 'private' stamp: 'm3r
setMinValue: min
    self error: 'minValue for a DigitAE is always 0!!'! !


!DigitCounterAE methodsFor: 'accessing' stamp: 'm3r
absoluteMaxValue
    ^(10 raisedToInteger: self digitCount) - 1! !

!DigitCounterAE methodsFor: 'accessing' stamp: 'm3r
digitCount
    ^self subelements size! !

!DigitCounterAE methodsFor: 'accessing' stamp: 'm3r
digitGapRatio
    ^self findPropertyAt: #digitGapRatio ifAbsentAtTopContextPut:
        [:topContext | self defaultDigitGapRatio]! !

!DigitCounterAE methodsFor: 'accessing' stamp: 'm3r
maxValue
    ^maxValue! !

!DigitCounterAE methodsFor: 'accessing' stamp: 'm3r
maxValue: max
    max < 0 ifTrue: [self error: 'Can''t handle negative numbers yet!!'].
    self setMaxValue: (max min: self absoluteMaxValue).
    self value > max ifTrue: [self value: max].! !

!DigitCounterAE methodsFor: 'accessing' stamp: 'm3r
minValue
    ^minValue! !

!DigitCounterAE methodsFor: 'accessing' stamp: 'm3r
minValue: min
    min < 0 ifTrue: [self error: 'Can''t handle negative numbers yet!!'].
    self setMinValue: (min min: self absoluteMaxValue).
    self value < min ifTrue: [self value: min].! !

!DigitCounterAE methodsFor: 'handling change' stamp: 'm3r
boundsChanged
    | digitCount gapCount newBounds width widthUnit digitWidth
      digitBounds totalDigitWidth gapWidth remaining origin dx x y |
    digitCount _ self digitCount.
    gapCount _ digitCount - 1.
    newBounds _ self getBounds.
    width _ newBounds width.
    widthUnit _ gapCount * self digitGapRatio + digitCount.

    digitWidth _ width // widthUnit.
    digitBounds _ Rectangle origin: 0@0 corner: (digitWidth @ newBounds height).

    totalDigitWidth _ digitWidth * digitCount.
    gapWidth _ width - totalDigitWidth // gapCount.
    remaining _ width - totalDigitWidth - (gapWidth * gapCount).

    origin _ newBounds origin.
    y _ origin y.
    x _ ((remaining / 2) + 0.499) truncated + origin x.
    dx _ digitWidth + gapWidth.

    super boundsChanged.
    self subelementsReverseDo:
        [:digit | digit
            setLocationBounds: (digitBounds moveSelfTo: (x @ y));
            boundsChanged.
        x _ x + dx].
    ^false
    ! !

!DigitCounterAE methodsFor: 'handling change' stamp: 'm3r
valueChangedFrom: previousValue
    | remaining range hideLeadingZeros notLeadingZeros |
    (self doAnimateValueChange: previousValue) ifFalse: [^false].
    self aspectChanged.

    hideLeadingZeros _ self isHidingLeadingZeros.
    notLeadingZeros _ true.
    remaining _ self value.
    self subelementsDo:
        [:element | element isCounter
```

```
        ifFalse: [element aspectChanged: #value with: previousValue]
        ifTrue: [notLeadingZeros
                ifFalse: [element setValue: #leadingZero]
                ifTrue: [range _ element range.
                      element setValue: remaining \\ range.
                      remaining _ remaining // range.
                      (hideLeadingZeros and: [remaining = 0])
                         ifTrue: [notLeadingZeros _ true]]]].
    ^false! !

!DigitCounterAE methodsFor: 'defaults' stamp: 'm3r
defaultDigitGapRatio
    ^0.125! !

!DigitCounterAE methodsFor: 'defaults'!
defaultSegmentLWratio
    ^4.0
! !

!DigitCounterAE methodsFor: 'defaults'!
defaultTipLWratio
    ^1.0
! !

!DigitCounterAE methodsFor: 'private' stamp: 'm3r
buildDigits: count ofType: aDigitClass
    self subelements: ((1 to: count) collect: [:index | aDigitClass new]).
    minValue _ 0.
    maxValue _ self absoluteMaxValue.
! !

!DigitCounterAE methodsFor: 'private' stamp: 'm3r
setMaxValue: max
    maxValue _ max.! !

!DigitCounterAE methodsFor: 'private' stamp: 'm3r
setMinValue: min
    minValue _ min.! !


!DigitCounterAE class reorganize!
('instance creation' new:width: new:width:in: width:)
('defaults' defaultDigitClass)
('xxx' digitCount: digits: new:digitCount: new:digitCount:in: new:digits: new:digits:in:)
!


!DigitCounterAE class methodsFor: 'instance creation' stamp: 'm3r
new: aDigitClass width: count
    ^self new buildDigits: count ofType: aDigitClass! !

!DigitCounterAE class methodsFor: 'instance creation' stamp: 'm3r
new: aDigitClass width: count in: anAnimationElement
    ^(self on: anAnimationElement) buildDigits: count ofType: aDigitClass! !

!DigitCounterAE class methodsFor: 'instance creation' stamp: 'm3r
width: count
    ^self new buildDigits: count ofType: self defaultDigitClass! !

!DigitCounterAE class methodsFor: 'defaults' stamp: 'm3r
defaultDigitClass
    ^FoldingDigitAE! !

!DigitCounterAE class methodsFor: 'xxx' stamp: 'm3r
digitCount: count
    ^self new buildDigits: count ofType: self defaultDigitClass! !

!DigitCounterAE class methodsFor: 'xxx' stamp: 'm3r
digits: count
    ^self new buildDigits: count ofType: self defaultDigitClass! !

!DigitCounterAE class methodsFor: 'xxx' stamp: 'm3r
new: aDigitClass digitCount: count
    ^self new buildDigits: count ofType: aDigitClass! !

!DigitCounterAE class methodsFor: 'xxx' stamp: 'm3r
new: aDigitClass digitCount: count in: anAnimationElement
    ^(self on: anAnimationElement) buildDigits: count ofType: aDigitClass! !

!DigitCounterAE class methodsFor: 'xxx' stamp: 'm3r
new: aDigitClass digits: count
    ^self new buildDigits: count ofType: aDigitClass! !

!DigitCounterAE class methodsFor: 'xxx' stamp: 'm3r
new: aDigitClass digits: count in: anAnimationElement
    ^(self on: anAnimationElement) buildDigits: count ofType: aDigitClass! !


!DisplayProfile reorganize!
('accessing' depth extent frameCount frameSequenceAt: frameSequences: height height: stepCount width w
('comparing' = calcHash hash)
('calculating extent' setExtentFrom:)
('private' initialize:)
!


!DisplayProfile methodsFor: 'accessing' stamp: 'm3r
depth
    ^depth! !

!DisplayProfile methodsFor: 'accessing' stamp: 'm3r
extent
    ^width @ height! !

!DisplayProfile methodsFor: 'accessing' stamp: 'm3r
frameCount
    ^frameCount! !

!DisplayProfile methodsFor: 'accessing' stamp: 'm3r
frameSequenceAt: key
    ^frameSequences at: key! !

!DisplayProfile methodsFor: 'accessing' stamp: 'm3r
frameSequences: aKeyedCollection
    frameSequences _ aKeyedCollection.! !

!DisplayProfile methodsFor: 'accessing' stamp: 'm3r
height
    ^height! !

!DisplayProfile methodsFor: 'accessing' stamp: 'm3r
height: aHeight
    height _ aHeight! !

!DisplayProfile methodsFor: 'accessing' stamp: 'm3r
stepCount
    ^frameCount - 1! !

!DisplayProfile methodsFor: 'accessing' stamp: 'm3r
width
    ^width! !

!DisplayProfile methodsFor: 'accessing' stamp: 'm3r
width: aWidth
    width _ aWidth! !
```

```
!DisplayProfile methodsFor: 'comparing' stamp: 'm3r
= aProfile
    width = aProfile width ifFalse: [^false].
    height = aProfile height ifFalse: [^false].
    depth = aProfile depth ifFalse: [^false].
    ^frameCount = aProfile frameCount! !

!DisplayProfile methodsFor: 'comparing' stamp: 'm3r
calcHash
    hash _ self class hash.
    {depth. width. height. frameCount} do:
        [:number | hash _ (hash bitShift: 2) bitXor: number].
    ^hash! !

!DisplayProfile methodsFor: 'comparing' stamp: 'm3r
hash
    ^hash ifNil: [self calcHash]
! !

!DisplayProfile methodsFor: 'calculating extent' stamp: 'm3r
setExtentFrom: anAnimationElement
    | extent elementWidth elementHeight hvFillRatio hvMinBorderGap |
    extent _ anAnimationElement extent.
    elementWidth _ extent x.
    elementHeight _ extent y.
    hvFillRatio _ anAnimationElement hvFillRatio.
    hvMinBorderGap _ anAnimationElement hvMinBorderGap * 2.

    width _ (width * hvFillRatio x) rounded.
    height _ (height * hvFillRatio y) rounded.
    width _ (elementWidth - width) >= hvMinBorderGap x
        ifTrue: [width] ifFalse: [width - hvMinBorderGap x].
    height _ (elementHeight - height) >= hvMinBorderGap y
        ifTrue: [height] ifFalse: [height - hvMinBorderGap y].
! !

!DisplayProfile methodsFor: 'private' stamp: 'm3r
initialize: anAnimationElement
    | extent |
    extent _ anAnimationElement internalExtent.
    width _ extent x.
    height _ extent y.
    depth _ anAnimationElement depth.
    frameCount _ anAnimationElement frameCount.! !


!DisplayProfile class reorganize!
('accessing' clearAllProfiles clearProfiles factoryClass profileFor:)
!


!DisplayProfile class methodsFor: 'accessing' stamp: 'm3r
clearAllProfiles
        "DisplayProfile clearAllProfiles."
    DisplayProfile allSubclassesDo:
        [:class | class clearProfiles]! !

!DisplayProfile class methodsFor: 'accessing' stamp: 'm3r
clearProfiles
    ^self factoryClass clearProfiles! !

!DisplayProfile class methodsFor: 'accessing' stamp: 'm3r
factoryClass
    self subclassResponsibility! !

!DisplayProfile class methodsFor: 'accessing' stamp: 'm3r
profileFor: anAnimationElement
    | newProfile |
    newProfile _ self new initialize: anAnimationElement.
    newProfile extent > (000) ifFalse: [^nil].
    ^self factoryClass buildOn: newProfile
! !

!FrameSequence reorganize!
('all' form formAt: forms isStatic points position positionAt: setPoints:forms:)
!


!FrameSequence methodsFor: 'all' stamp: 'm3r
form
    ^formSteps = 0
        ifTrue: [forms]
        ifFalse: [forms at: 1]
! !

!FrameSequence methodsFor: 'all' stamp: 'm3r
formAt: aNumber
    ^formSteps = 0
        ifTrue: [forms]
        ifFalse: [forms at: (aNumber asUnitIndexIn: formSteps) + 1]
! !

!FrameSequence methodsFor: 'all' stamp: 'm3r
forms
    ^formSteps = 0
        ifTrue: [Array with: forms]
        ifFalse: [forms]
! !

!FrameSequence methodsFor: 'all' stamp: 'm3r
isStatic
    ^pointSteps = 0 and: [formSteps = 0]! !

!FrameSequence methodsFor: 'all' stamp: 'm3r'
points
    ^pointSteps = 0
        ifTrue: [Array with: points]
        ifFalse: [points]! !

!FrameSequence methodsFor: 'all' stamp: 'm3r
position
    ^pointSteps = 0
        ifTrue: [points]
        ifFalse: [points at: 1]! !

!FrameSequence methodsFor: 'all' stamp: 'm3r'
positionAt: aNumber
    ^pointSteps = 0
        ifTrue: [points]
        ifFalse: [points at: (aNumber asUnitIndexIn: pointSteps) + 1]
! !

!FrameSequence methodsFor: 'all' stamp: 'm3r
setPoints: pointsArray forms: formsArray
    points _ (pointSteps _ pointsArray size - 1) = 0
        ifTrue: [pointsArray at: 1] ifFalse: [pointsArray].
    forms _ (formSteps _ formsArray size - 1) = 0
        ifTrue: [formsArray at: 1] ifFalse: [formsArray].! !


!FrameSequence class methodsFor: 'instance creation' stamp: 'm3r
points: pointsArray forms: formsArray
    ^self new setPoints: pointsArray forms: formsArray! !


!Number methodsFor: 'Numericon support' stamp: 'm3r
```

```
asUnitIndexIn: anIntegerRange
    self subclassResponsibility.! !


!Float methodsFor: 'Numericon support' stamp: 'm3r
asUnitIndexIn: anIntegerRange
    (self < 0.0 or: [self > 1.0]) ifTrue: [self error: 'Unit index of range!!'].
    ^(self * anIntegerRange) rounded! !


!Fraction methodsFor: 'Numericon support' stamp: 'm3r
asUnitIndexIn: anIntegerRange
    ^(denominator = anIntegerRange
        ifTrue: [numerator] ifFalse: [self asFloat])
            asUnitIndexIn: anIntegerRange
! !

!Fraction methodsFor: 'Numericon support' stamp: 'm3r
decrement
    numerator _ numerator - 1.! !

!Fraction methodsFor: 'Numericon support' stamp: 'm3r
increment
    numerator _ numerator + 1.! !


!Integer methodsFor: 'Numericon support' stamp: 'm3r
asUnitIndexIn: anIntegerRange
    (self < 0 or: [self > anIntegerRange]) ifTrue: [self error: 'Unit index of range!!'].
    ^self! !
```

```
'From Squeak 2.3                                    !
DisplayProfile variableSubclass: #FDDisplayProfile
    instanceVariableNames: 'segmentLWratio tipLWratio '
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Numerican-FoldingDigits'!
DisplayProfileFactory subclass: #FDDisplayProfileFactory
    instanceVariableNames: 'anchorLength segmentLength segmentWidth tipGap tipLe
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Numerican-FoldingDigits'!
AnimationElement subclass: #FDSegmentAE
    instanceVariableNames: 'positionMove anchor sequence '
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Numerican-FoldingDigits'!
DigitAE subclass: #FoldingDigitAE
    instanceVariableNames: 'transition displayProfile staticGraphics compositeGr
    classVariableNames: 'TransitionsTable '
    poolDictionaries: ''
    category: 'Numerican-FoldingDigits'!

!FDDisplayProfile methodsFor: 'accessing' stamp: 'm3r
anchorSequencePairAt: aPositionMove
    ^self frameSequenceAt: aPositionMove.! !

!FDDisplayProfile methodsFor: 'accessing' stamp: 'm3r.
anchorSequencePairs: anAssociation
    self frameSequences: anAssociation.! !

!FDDisplayProfile methodsFor: 'accessing' stamp: 'm3r
segmentFillRatio
    ^self defaultSegmentFillRatio! !

!FDDisplayProfile methodsFor: 'accessing' stamp: 'm3r
segmentLWratio
    ^segmentLWratio! !

!FDDisplayProfile methodsFor: 'accessing' stamp: 'm3r
tipLWratio
    ^tipLWratio! !

!FDDisplayProfile methodsFor: 'comparing' stamp: 'm3r .
= aProfile
    super = aProfile ifFalse: [^false].
    segmentLWratio = aProfile segmentLWratio ifFalse: [^false].
    ^tipLWratio = aProfile tipLWratio
! !

!FDDisplayProfile methodsFor: 'comparing' stamp: 'm3r
calcHash
    hash _ super calcHash.
    {segmentLWratio. tipLWratio} do:
        [:ratio | hash _ (hash bitShift: 2) bitXor: ratio hash].
    ^hash! !

!FDDisplayProfile methodsFor: 'defaults' stamp: 'm3r
defaultSegmentFillRatio
    ^0.82842712474619

    "| distance slipCenter delta gap |
    distance _ 2 sqrt / 2.
    slipCenter _ 1 - (2 sqrt / 4).
    delta _ distance - slipCenter.
    gap _ (2 * delta squared) sqrt.
    ^1 - (2 * gap) "
! !

!FDDisplayProfile methodsFor: 'private' stamp: 'm3r
initialize: aDigitCounter
    super initialize: aDigitCounter.
    segmentLWratio _ aDigitCounter segmentLWratio.
    tipLWratio _ aDigitCounter tipLWratio.
    ! !


!FDDisplayProfile class methodsFor: 'accessing' stamp: 'm3r
factoryClass
    ^FDDisplayProfileFactory
! !

Smalltalk renameClassNamed: #FDDPFactory as: #FDDisplayProfileFactory!

!FDDisplayProfileFactory reorganize!
('building' build calcAnchorPositions calcFrameSequences calcOffsetsLists calcR
('calculating extent' calcExtent calcHeightFromConstrainedWidth: calcWidthFromC
('calculating segment forms' calcSegmentForm: calcTriangleForm:)
('storing sequences' storeSequencesFor:points:forms: storeUniqueRotations store
('old' calcCanonicalSegmentForm: calcSegmentForm:depth: calcSmoothTriangleForm:
!


!FDDisplayProfileFactory methodsFor: 'building' stamp: 'm3r
build
    ^self
        calcSegmentDimensions;
        calcAnchorPositions;
        calcRotationPath;
        calcSegmentForms;
        calcOffsetsLists;
        calcFrameSequences;
        displayProfile! !

!FDDisplayProfileFactory methodsFor: 'building' stamp: 'm3r
calcAnchorPositions
    | halfAnchorLength topEdge leftEdge centerX rightEdge
     midTop middle midBottom bottomEdge |
    halfAnchorLength _ anchorLength / 2.

    topEdge _ leftEdge _ segmentWidth / 2.
    centerX _ leftEdge + halfAnchorLength.
    rightEdge _ leftEdge + anchorLength.

    midTop _ topEdge + halfAnchorLength.
    middle _ topEdge + anchorLength.
    midBottom _ middle + halfAnchorLength.
    bottomEdge _ middle + anchorLength.

    (namedAnchors _ IdentityDictionary new: 15)
        at: #A put: (centerX @ topEdge) truncated;
        at: #B put: (rightEdge @ midTop) truncated;
        at: #C put: (rightEdge @ midBottom) truncated;
        at: #D put: (centerX @ bottomEdge) truncated;
        at: #E put: (leftEdge @ midBottom) truncated;
        at: #F put: (leftEdge @ midTop) truncated;
        at: #G put: (centerX @ middle) truncated;
        at: #Gu put: (centerX @ middle) truncated;
        at: #Gd put: (centerX @ middle) truncated;
        at: #topLeft put: (leftEdge @ topEdge) truncated;
        at: #topRight put: (rightEdge @ topEdge) truncated;
        at: #middleLeft put: (leftEdge @ middle) truncated;
        at: #middleRight put: (rightEdge @ middle) truncated;
        at: #bottomLeft put: (leftEdge @ bottomEdge) truncated;
        at: #bottomRight put: (rightEdge @ bottomEdge) truncated;
        yourself.

    segmentAnchors _ #(topLeft topRight middleRight middleLeft
                    middleLeft middleRight bottomRight bottomLeft)
                        collect: [:key | namedAnchors at: key].! !

!FDDisplayProfileFactory methodsFor: 'building' stamp: 'm3r
calcFrameSequences
    | stepCount startIndex dynamicFormsLists staticFormsLists zeroList |
    stepCount _ profile stepCount.
    anchorSequencePairs _ IdentityDictionary new: 44.
        "8 segments * 5 movement types + 4 special moves"

    dynamicFormsLists _ (0 to: 3) collect:
        [:quadrant | startIndex _ quadrant * stepCount + 1.
        forms copyFrom: startIndex to: startIndex + stepCount].
    staticFormsLists _ dynamicFormsLists collect: [:list | list copyFrom: 1 to: :
    zeroList _ {{0@0}}.

    self
        storeSequencesFor: 'rotateFrom' points: zeroList forms: dynamicFormsLists;
        storeSequencesFor: 'slipFrom' points: deceleratingRDLU forms: dynamicForm
        storeSequencesFor: 'pushFrom' points: deceleratingDLUR forms: staticForms
        storeSequencesFor: 'pullFrom' points: acceleratingDLUR forms: staticForms
        storeSequencesFor: 'moveFrom' points: linearDLUR forms: staticFormsLists;
        storeSequencesFor: #static points: zeroList forms: staticFormsLists;
        storeUniqueSpins;
        storeUniqueRotations;
        yourself.

    profile anchorSequencePairs: anchorSequencePairs.! !

!FDDisplayProfileFactory methodsFor: 'building' stamp: 'm3r
calcOffsetsLists
    | frameCount accelerating decelerating linear form anchor steps directions |

    frameCount _ profile frameCount.
    accelerating _ Array new: frameCount.
    decelerating _ Array new: frameCount.
    linear _ Array new: frameCount.
    steps _ frameCount - 1.
```

```
    1 to: frameCount do:
        [:index | form _ forms at: index.
        anchor _ form extent * (form offset * 2) - (101).
        accelerating at: index put: (anchorLength * anchor x).
        decelerating at: index put: anchor y.
        linear at: index put: (anchorLength * (index - 1) / steps) rounded].

    directions _ {001. -100. 00-1. 100}.
    acceleratingDLUR _ directions collect: [:direction | direction * acceleratin
    deceleratingDLUR _ directions collect: [:direction | direction * deceleratin
    linearDLUR _ directions collect: [:direction | direction * linear].

    decceleratingRDLU _ Array new: decceleratingDLUR size.
    decceleratingRDLU atAll: #(2 3 4 1) putAll: decceleratingDLUR.! !

!FDDisplayProfileFactory methodsFor: 'building' stamp: 'm3r
calcRotationPath
    | y2xArc stepCount frameCount y2xArcPoints stream point offset |
    y2xArc _ Arc new center: 0.000.0 radius: anchorLength quadrant: 4.
    stepCount _ profile stepCount.
    frameCount _ stepCount + 1.
    y2xArcPoints _ (y2xArc asLinearFit: frameCount) points.

    pathPoints _ Array new: (stepCount * 4).
    stream _ WriteStream on: pathPoints.
    frameCount to: 2 by: -1 do:
        [:index | stream nextPut: (y2xArcPoints at: index)].
    1 to: stepCount do:
        [:index | point _ (y2xArcPoints at: index).
        stream nextPut: (0 - point x) @ (point y)].
    offset _ stepCount * 2.
    1 to: offset do:
        [:index | point _ pathPoints at: index.
        pathPoints
            at: index + offset
            put: (0 - point x) @ (0 - point y)].! !

!FDDisplayProfileFactory methodsFor: 'building' stamp: 'm3r
calcSegmentDimensions
    tipLength _ (segmentWidth * profile tipLWratio) rounded.
    tipGap _ (anchorLength * (1 - profile segmentFillRatio) / 2) truncated + 1.
    segmentLength _ anchorLength - (2 * tipGap) - 1.! !

!FDDisplayProfileFactory methodsFor: 'building' stamp: 'm3r
calcSegmentForms
    | baseAngle scale uniqueCount totalCount angle
     canonicalSegmentForm rotatedForm segmentForm |
    baseAngle _ 90.0 / profile stepCount.
    scale _ profile depth.
    canonicalSegmentForm _ self calcSegmentForm: scale.
    uniqueCount _ 2 * profile stepCount.
    totalCount _ uniqueCount * 2 + 1.
    forms _ Array new: totalCount.

    1 to: uniqueCount do:
        [:index | angle _ baseAngle * (index - 1).
        segmentForm _ scale = 1
            ifTrue: [rotatedForm _ canonicalSegmentForm
                    rotateBy: angle magnify: (1 / smoothingScale) smoothing: 2.
                rotatedForm trimToPixelValue: 1 orNot: false.]
            ifFalse: [rotatedForm _ canonicalSegmentForm
                    rotateBy: angle smoothing: 1.
                rotatedForm _ rotatedForm trimToPixelValue: 1 orNot: false.
                rotatedForm shrinkAndSmoothBy: scale].
        forms
            at: index put: segmentForm;
            at: index + uniqueCount put: segmentForm copy].

    pathPoints withIndexDo:
        [:anchor :index | segmentForm _ forms at: index.
        segmentForm offset: ((anchor - segmentForm extent) / 2) ceiling].
    forms at: totalCount put: (forms at: 1).! !

!FDDisplayProfileFactory methodsFor: 'calculating extent' stamp: 'm3r
calcExtent
    | c absoluteVHratio internalVHratio |

    c _ profile segmentLWratio / profile segmentFillRatio.
    absoluteVHratio _ (2 * c + 1) / (c + 1).
    internalVHratio _ (profile height / profile width) asFloat.
    internalVHratio > absoluteVHratio
        ifTrue: [self calcHeightFromConstrainedWidth: c]
        ifFalse: [self calcWidthFromConstrainedHeight: c].! !

!FDDisplayProfileFactory methodsFor: 'calculating extent' stamp: 'm3r
calcHeightFromConstrainedWidth: ratioConst
    | width newHeight delta |
    width _ profile width.
    segmentWidth _ width / (ratioConst + 1).
    segmentWidth _ segmentWidth roundToOdd.
    anchorLength _ width - segmentWidth.
    newHeight _ 2 * anchorLength + segmentWidth.
    (delta _ newHeight - profile height) > 0 ifTrue:
        [anchorLength _ anchorLength - delta.
        newHeight _ 2 * anchorLength + segmentWidth].
    profile height: newHeight.
! !

!FDDisplayProfileFactory methodsFor: 'calculating extent' stamp: 'm3r
calcWidthFromConstrainedHeight: ratioConst
    | height newWidth delta |
    height _ profile height.
    segmentWidth _ height / (2 * ratioConst + 1).
    segmentWidth _ segmentWidth roundToOdd.
    anchorLength _ ((height - segmentWidth) / 2) truncated.
    newWidth _ anchorLength + segmentWidth.
    (delta _ newWidth - profile width) > 0 ifTrue:
        [anchorLength _ anchorLength - delta.
        newWidth _ anchorLength + segmentWidth].
    profile width: newWidth.! !

!FDDisplayProfileFactory methodsFor: 'calculating segment forms' stamp: 'm3r
calcSegmentForm: scale
    | segHalfWidth segWidth segLength segTipLength
     topRightTriangleForm topLeftTriangleForm
     bottomRightTriangleForm bottomLeftTriangleForm
     rightTipBase |
    segWidth _ segmentWidth * scale.
    segHalfWidth _ segWidth / 2.
    segLength _ segmentLength * scale.
    segTipLength _ tipLength * scale.

    topRightTriangleForm _ self calcTriangleForm: segTipLength @ segHalfWidth.
    topLeftTriangleForm _ topRightTriangleForm flipBy: #horizontal.
    bottomRightTriangleForm _ topRightTriangleForm flipBy: #vertical.
    bottomLeftTriangleForm _ topLeftTriangleForm flipBy: #vertical.

    rightTipBase _ segLength - segTipLength.
    ^(GraphicsContext on: (Form extent: segLength @ segWidth))
        fillBlack;
        display: topLefeTriangleForm at: 000;
        display: bottomLeftTriangleForm at: 0 @ segHalfWidth;
        display: topRightTriangleForm at: rightTipBase @ 0;
        display: bottomRightTriangleForm at: rightTipBase @ segHalfWidth;
        form! !

!FDDisplayProfileFactory methodsFor: 'calculating segment forms' stamp: 'm3r
calcTriangleForm: extent
    | g ratio line x |
    g _ GraphicsContext extent: extent rounded depth: 1.
    line _ Line new.
    ratio _ extent x / extent y.
    0 to: extent y - 1 do:
        [:y | x _ (y * ratio) rounded + 1.
        line from: 0 @ y to: x @ y.
        g display: line].
    ^g form! !

!FDDisplayProfileFactory methodsFor: 'storing sequences' stamp: 'm3r
storeSequencesFor: action points: pointsLists forms: formsLists
    FoldingDigitAE segmentNames withIndexDo:
        [:segmentName :index | anchorSequencePairs
            at: (action == #static
                    ifTrue: [(segmentName at: 1) asSymbol]
                    ifFalse: [(action , segmentName) asSymbol])
                put: (segmentAnchors at: index) ->
                    (FrameSequence
                        points: (pointsLists atWrap: index)
                        forms: (formsLists atWrap: index))].
! !

!FDDisplayProfileFactory methodsFor: 'storing sequences' stamp: 'm3r
storeUniqueRotations
    | stepCount frameCount wideRotationForms decceleratingUpOffsets horizontalFo
     rotateAndPushOffsets rotateAndPushForms pointsStream formsStream middleRi
    stepCount _ profile stepCount.
    wideRotationForms _ forms atAll: (stepCount + 1 to: stepCount * 3 + 1 by: 2).
    middleRight _ namedAnchors at: #middleRight.

    anchorSequencePairs at: #wideRotateC put:
        middleRight -> (FrameSequence points: {000} forms: wideRotationForms).

    frameCount _ stepCount + 1.
    rotateAndPushOffsets _ Array new: frameCount.
    rotateAndPushForms _ Array new: frameCount.
    pointsStream _ WriteStream on: rotateAndPushOffsets.
```

```
    formsStream _ WriteStream on: rotateAndPushForms.
    decelerateingUpOffsets _ decelerateingDLUR at: 3.
    horizontalForm _ forms at: stepCount * 2 + 1.

    1 to: frameCount // 2 do:
        [:index | pointsStream nextPut: 000.
        formsStream nextPut: (wideRotationForms at: index)].
    (frameCount even ifFalse: [1] ifTrue: [2]) to: frameCount by: 2 do:
        [:index | pointsStream nextPut: (decelerateingUpOffsets at: index).
        formsStream nextPut: horizontalForm].

    anchorSequencePairs at: #rotateAndPushC put:
        middleRight ->
            (FrameSequence points: rotateAndPushOffsets forms: rotateAndPushForms

!FDDisplayProfileFactory methodsFor: 'storing sequences' stamp: 'm3r
storeUniqueSpins
    | stepCount spinForms form |
    stepCount _ profile stepCount.
    spinForms _ (1 to: stepCount * 2 + 1 by: 2) collect:
        [:index | form _ forms at: index.
        form copy offset: (0 - (form extent // 2))].

    anchorSequencePairs at: #spinG put:
        (namedAnchors at: #G) -> (FrameSequence points: {000} forms: spinForms).

    spinForms _ (stepCount + 1 to: stepCount * 3 + 1 by: 2) collect:
        [:index | form _ forms at: index.
        form copy offset: (0 - (form extent // 2))].

    anchorSequencePairs at: #diagonalSpinE put:
        (namedAnchors at: #E) ->
            (FrameSequence
                points: (linearDLUR at: 4) + (linearDLUR at: 3)
                forms: spinForms).! !

!FDDisplayProfileFactory methodsFor: 'old' stamp: 'm3r
calcCanonicalSegmentForm: scale
    | left right bottom |
    left _ tipLength * scale.
    right _ (segmentLength - tipLength) * scale.
    bottom _ segmentWidth * scale - 1.

    canonicalSegmentForm _ self calcSegmentForm: scale depth: 4.
    (GraphicsContext on: canonicalSegmentForm)
        maskPattern: (Bitmap with: 16r31313131);
        drawLineFrom: (left @ 0) to: (right @ 0);
        maskPattern: (Bitmap with: 16r13131313);
        drawLineFrom: (left - 1 @ bottom) to: (right - 1 @ bottom).


! !

!FDDisplayProfileFactory methodsFor: 'old' stamp: 'm3r
calcSegmentForm: scale depth: depth
    | segHalfWidth segWidth segLength segTipLength
     topRightTriangleForm topLeftTriangleForm
     bottomRightTriangleForm bottomLeftTriangleForm
     bounds rightTipBase g |
    segHalfWidth _ segmentWidth * scale.
    segWidth _ segHalfWidth * 2.
    segLength _ segmentLength * scale * 2.
    segTipLength _ tipLength * scale * 2.

    topRightTriangleForm _ self calcTriangleForm: segTipLength @ segHalfWidth.
    topLeftTriangleForm _ topRightTriangleForm flipBy: #horizontal.
    bottomRightTriangleForm _ topRightTriangleForm flipBy: #vertical.
    bottomLeftTriangleForm _ topLeftTriangleForm flipBy: #vertical.

    rightTipBase _ segLength - segTipLength.
    bounds _ 000 corner: segLength @ segWidth.
    (g _ GraphicsContext bounds: bounds depth: depth)
        fillBlack;
        display: topLeftTriangleForm at: 000;
        display: bottomLeftTriangleForm at: 0 @ segHalfWidth;
        display: topRightTriangleForm at: rightTipBase @ 0;
        display: bottomRightTriangleForm at: rightTipBase @ segHalfWidth.

    ^g form magnify: bounds by: 0.5 smoothing: (depth > 1 ifTrue: [1] ifFalse: [2
! !

!FDDisplayProfileFactory methodsFor: 'old' stamp: 'm3r
calcSmoothTriangleForm: extent
    | g ratio line x scaledExtent |
    scaledExtent _ extent * 4.
    g _ GraphicsContext extent: scaledExtent depth: 1.
    line _ Line new.
    ratio _ scaledExtent x / scaledExtent y.
    0 to: scaledExtent y - 1 do:
        [:y | x _ (y * ratio) rounded + 1.
        line from: 0 @ y to: x @ y.
        g display: line].
    ^g form shrinkBy: 4 smoothToDepth: 4.! !


!FDDisplayProfileFactory class methodsFor: 'building' stamp: 'm3r
buildOn: rawProfile
    | factory |
    (factory _ self new)
        setDisplayProfile: rawProfile;
        calcExtent.
    ^self profileAt: factory displayProfile ifAbsentPut: [factory build]
! !


!FDSegmentAE reorganize!
('all' deactivate isDeactivated positionMove positionMove: setAnchorAndSequence
('area calculation' drawingArea)
!


!FDSegmentAE methodsFor: 'all' stamp: 'm3r
deactivate
    anchor _ sequence _ nil.
    self
        setLocation: 000;
        setBounds: (000 corner: 000).
! !

!FDSegmentAE methodsFor: 'all' stamp: 'm3r
isDeactivated
    ^positionMove == nil! !

!FDSegmentAE methodsFor: 'all' stamp: 'm3r
positionMove
    ^positionMove! !

!FDSegmentAE methodsFor: 'all' stamp: 'm3r
positionMove: aSymbol
    positionMove _ aSymbol.! !

!FDSegmentAE methodsFor: 'all' stamp: 'm3r
setAnchorAndSequenceFrom: displayProfile
    | assoc index form |
    displayProfile == nil ifTrue: [^self deactivate].
    assoc _ displayProfile anchorSequencePairAt: positionMove.
    anchor _ assoc key.
    sequence _ assoc value.
    index _ self step.
    form _ sequence formAt: index.
    self
        setLocation: anchor + (sequence positionAt: index);
        setBounds: (form offset extent: form extent).
! !

!FDSegmentAE methodsFor: 'area calculation' stamp: 'm3r
drawingArea
    ^self basicDrawingArea! !


!FoldingDigitAE reorganize!
('accessing' colorMaps transition:)
('enumerating' activeSegmentsDo: dynamicSegmentsDo: staticSegmentsDo:)
('display profiling' depth depth:)
('handling change' aspectChanged boundsChanged colorChanged valueChangedFrom:)
('updating' assignSegments ensureDisplayProfile updateStaticForms)
('area calculation' drawingArea)
('private' assignDynamicSegments: assignStaticSegments: buildSegments initializ
('defaults' defaultDepth)
!


!FoldingDigitAE methodsFor: 'accessing' stamp: 'm3r
colorMaps
    ^colorMaps! !

!FoldingDigitAE methodsFor: 'accessing' stamp: 'm3r
transition: aTransition
    transition _ aTransition.
    ^self
        assignSegments;
        aspectChanged! !

!FoldingDigitAE methodsFor: 'enumerating' stamp: 'm3r
activeSegmentsDo: aOneArgBlock
    self subelementsDo:
```

```
        [:segment | segment positionMove == nil ifFalse:
            [aOneArgBlock value: segment]].! !

!FoldingDigitAE methodsFor: 'enumerating' stamp: 'm3r
dynamicSegmentsDo: aOneArgBlock
    self subelementsDo:
        [:segment | segment positionMove size > 1 ifFalse: [^self].
        aOneArgBlock value: segment].! !

!FoldingDigitAE methodsFor: 'enumerating' stamp: 'm3r
staticSegmentsDo: aOneArgBlock
    self subelementsReverseDo:
        [:segment | segment positionMove size = 1 ifFalse: [^self].
        aOneArgBlock value: segment].! !

!FoldingDigitAE methodsFor: 'display profiling' stamp: 'm3r
depth
    | context depth |
    ^self propertyAt: #depth ifAbsent:
        [(context _ self context) == nil
            ifTrue: [self propertyAt: #depth put: self defaultDepth]
            ifFalse: [depth _ context findPropertyAt: #depth ifAbsent:
                    [^self defaultDepth].
                depth min: self defaultDepth]]! !

!FoldingDigitAE methodsFor: 'display profiling' stamp: 'm3r
depth: newDepth
    newDepth > 4 ifTrue:
        [self error: 'FoldingDigits only supports depths of 1,2,4 bits!!'].
    super depth: newDepth.! !

!FoldingDigitAE methodsFor: 'handling change' stamp: 'm3r
aspectChanged
    self
        updateStaticForm;
        markVisibleAreaDirty.
    ^false! !

!FoldingDigitAE methodsFor: 'handling change' stamp: 'm3r
boundsChanged
    staticGraphics _ GraphicsContext extent: self extent depth: self depth.
    compositeGraphics _ nil "(Form extent: extent depth: digit graphics depth)".
    ^super boundsChanged! !

!FoldingDigitAE methodsFor: 'handling change' stamp: 'm3r
colorChanged
    colorMaps _ self class colorMapsFor: self color at: self depth.
    ^self aspectChanged! !

!FoldingDigitAE methodsFor: 'handling change' stamp: 'm3r
valueChangedFrom: previousValue
    self transition:
        ((self doAnimateValueChange: previousValue)
            ifFalse: [{self value}]
            ifTrue: [{transition last. self value}]).
    ^false! !

!FoldingDigitAE methodsFor: 'updating' stamp: 'm3r
assignSegments
    | actions |
    actions _ ReadStream on: (self class segmentActionsFor: transition).
    self
        assignStaticSegments: actions next;
        assignDynamicSegments: actions.! !

!FoldingDigitAE methodsFor: 'updating' stamp: 'm3r
ensureDisplayProfile
    displayProfile _ FDDisplayProfile profileFor: self.
    self activeSegmentsDo:
        [:segment | segment setAnchorAndSequenceFrom: displayProfile].
! !

!FoldingDigitAE methodsFor: 'updating' stamp: 'm3r
updateStaticForm
    staticGraphics == nil ifTrue: [^self].
    compositeGraphics _ nil.
    staticGraphics fillTransparent.
    self staticSegmentsDo:
        [:segment | segment drawOn: staticGraphics]! !

!FoldingDigitAE methodsFor: 'area calculation' stamp: 'm3r
drawingArea
    ^self basicDrawingArea! !

!FoldingDigitAE methodsFor: 'private' stamp: 'm3r
assignDynamicSegments: actionsStream
    self subelementsDo:
        [:element | actionsStream atEnd ifTrue: [^self].
        element
            positionMove: actionsStream next;
            setAnchorAndSequenceFrom: displayProfile]! !

!FoldingDigitAE methodsFor: 'private' stamp: 'm3r
assignStaticSegments: anArray
    | positions |
    positions _ ReadStream on: anArray.
    self subelementsReverseDo:
        [:element | positions atEnd
            ifTrue: [element
                    positionMove: nil;
                    deactivate]
            ifFalse: [element
                    positionMove: positions next;
                    setAnchorAndSequenceFrom: displayProfile]]! !

!FoldingDigitAE methodsFor: 'private' stamp: 'm3r
buildSegments
    ^self subelements:
        ((1 to: self class segmentNames size) collect: [:index | FDSegmentAE new])

!FoldingDigitAE methodsFor: 'private' stamp: 'm3r
initialize
    super initialize.
    transition _ Array new: 1.
    self buildSegments.! !

!FoldingDigitAE methodsFor: 'defaults' stamp: 'm3r
defaultDepth
    ^4! !


!FoldingDigitAE class reorganize!
('accessing' clearProfiles profileClass segmentActionsFor: segmentNames)
('reading in transitions' readInActionTypeFrom: readInActionTypesFrom: readInCe
('class initialization' buildSameToSameTransitions buildTransitionsTable ensure
!


!FoldingDigitAE class methodsFor: 'accessing' stamp: 'm3r
clearProfiles
    ^self profileClass clearProfiles! !

!FoldingDigitAE class methodsFor: 'accessing' stamp: 'm3r
profileClass
    ^FDDisplayProfile! !

!FoldingDigitAE class methodsFor: 'accessing' stamp: 'm3r
segmentActionsFor: anArray
    | from to |
    from _ (from _ anArray at: 1) == nil
        ifTrue: [11] ifFalse: [from + 1].
    anArray size = 1 ifTrue: [^TransitionsTable at: from].
    to _ (to _ anArray at: 2) == nil
        ifTrue: [11] ifFalse: [to + 1].
    ^TransitionsTable at: (from * 11 + to)! !

!FoldingDigitAE class methodsFor: 'accessing' stamp: 'm3r
segmentNames
    ^#(A B C D E F G)! !

!FoldingDigitAE class methodsFor: 'reading in transitions' stamp: 'm3r
readInActionTypeFrom: stream
    ^(String streamContents:
        [:actionName | actionName nextPut: stream next.
        [stream atEnd not and: [stream peek isLowercase]] whileTrue:
            [actionName nextPut: stream next]])
        caseOf: {['Ph'] -> ['pushFrom'].
                ['Pl'] -> ['pullFrom'].
                ['M'] -> ['moveFrom'].
                ['S'] -> ['slipFrom'].
                ['R'] -> ['rotateFrom']}! !

!FoldingDigitAE class methodsFor: 'reading in transitions' stamp: 'm3r'
readInActionTypesFrom: inStream
    ^Array streamContents:
        [:outStream | [inStream atEnd] whileFalse:
            [outStream nextPut: (self readInActionTypeFrom: inStream)]]! !

!FoldingDigitAE class methodsFor: 'reading in transitions' stamp: 'm3r'
readInCellFrom: stream
    ^stream upTo: Character tab.
! !

!FoldingDigitAE class methodsFor: 'reading in transitions' stamp: 'm3r
readInLineFrom: stream
    | transitionIndex segmentActions staticSegments segmentData |
```

```
    transitionIndex _ self readInTransitionKeyFrom: stream.
    segmentActions _ self readInSegmentsActionsFrom: stream.
    staticSegments _ self readInStaticSegmentsFrom: stream.

    segmentData _ segmentActions copyWithFirst: staticSegments.
    TransitionsTable at: transitionIndex put: segmentData.! !

!FoldingDigitAE class methodsFor: 'reading in transitions' stamp: 'm3r
readInSegmentActionsFrom: stream
    | segmentActions |
    ^(segmentActions _ self readInCellFrom: stream) size = 0
        ifTrue: [#()]
        ifFalse: [self readInActionTypesFrom: (ReadStream on: segmentActions)]! !

!FoldingDigitAE class methodsFor: 'reading in transitions' stamp: 'm3r
readInSegmentsActionsFrom: stream
    | actionWords |
    ^Array streamContents:
        [:outStream | #(A B C D E F Gu Gd) do:
            [:segmentName | actionWords _ self readInSegmentActionsFrom: stream.
            actionWords do:
                [:actionWord | outStream nextPut:
                    (actionWord , segmentName) asSymbol]]]! !

!FoldingDigitAE class methodsFor: 'reading in transitions' stamp: 'm3r
readInStaticSegmentsFrom: stream
    ^(stream upTo: Character cr) withBlanksTrimmed asArray collect:
        [:segment | segment asSymbol]
! !

!FoldingDigitAE class methodsFor: 'reading in transitions' stamp: 'm3r
readInTransitionKeyFrom: stream
    | from to |
    from _ self readInCellFrom: stream.
    from _ from size = 0 ifTrue: [11] ifFalse: [from asNumber + 1].
    to _ self readInCellFrom: stream.
    to _ to size = 0 ifTrue: [21] ifFalse: [to asNumber + 1].
    ^from - to ifTrue: [to] ifFalse: [from * 11 + to]! !

!FoldingDigitAE class methodsFor: 'reading in transitions' stamp: 'm3r
readInTransitionsData
    | stream |
    (stream _ ReadStream on: self transitionsData)
        upTo: Character cr.
    [stream atEnd] whileFalse:
        [self readInLineFrom: stream].! !

!FoldingDigitAE class methodsFor: 'reading in transitions' stamp: 'm3r
transitionsData
^'    A  B  C  D  E  F  Gu Gd Static
   0                    PhRS  PhRS
   1                    R  S
   2                    PhR PhR G
   3                    PhR PhS G
   4                    RS  S  G
   5                    PhS PhS G
   6                    PhS PhRS  G
   7                    PhRS
   8                    PhRS  PhRS  G
   9                    PhRS  PhS G
0     Pl S  R  Pl S  R                 G
1        S  R
2     Pl S     Pl S                    G
3     Pl S  R  Pl                      G
4        S  R        R                 G
5     Pl    R  Pl    R                 G
6     Pl    R  Pl S  R                 G
7     Pl S  R
8     Pl S  R  Pl S  R                 G
9     Pl S  R  Pl    R                 G
0 0                                    ABCDEF
0 1 S          R  Pl Pl                BC
0 2         R        R                 ABDE
0 3                  S  R              ABCD
0 4 Pl          Pl S                   BCF
0 5    S        S                      ACDF
0 6    S                               ACDEF
0 7              R  Pl Pl              ABC
0 8 M            M                     ABCDEF
0 9                 S                  ABCDF
1 0       PhR PhS                      BC
1 1                                    BC
1 2       RS  PhS                      B
1 3       RS  S                        BC
1 4       PhS                          BC
1 5       PhRS  S                         C
1 6       PhRS  PhS                       C
1 7       R                            BC
1 8       PhRS  PhS                    BC
1 9       PhRS  S                      BC
2 0                          S  S  ABDE
2 1 S          R  Pl     R        B
2 2                                  ABDEG
2 9                   M              ABDG
2 4 R          R  S                  BG
2 5    M          M                  ADG
2 6 R  S                       S     ADE
2 7              R  Pl     R        AB
2 8 R            R                  ABDEG
2 9              R  S      S        ABD
3 0                        S  R     ABCD
3 1 S            R        R        BC
3 2       M                         ABDG
3 3                                  ABCDG
3 4 R            R                  BCG
3 5    M                             ACDG
3 6 R  S                       R     ACD
3 7              R        R        ABC
3 8 R            S                 ABCDG
3 9 R                              ABCDG
4 0                          Ph PhR BCF
4 1                    Pl       S  BC
4 2          S         S        R  BG
4 3          S         S           BCG
4 4                                BCFG
4 5    R  S                       CFG
4 6    R  S                    R  CFG
4 7                    S        S  BC
4 8                          Ph PhR BCFG
4 9                          M  M  BCFG
5 0                          R  R  ACDF
5 1 S          R        Pl R        C
5 2       M              M           ADG
5 3                       M           ACDG
5 4 S          R                     CFG
5 5                                  ACDFG
5 6                                R ACDFG
5 7              R     S  R        AC
5 8                       R  R     ACDFG
5 9                       R        ACDFG
6 0                       R        ACDEF
6 1 S          R  Pl Pl R           C
6 2       R              S  R        ADE
6 3                    S  S  R     ACD
6 4 S          R  S                 CFG
6 5              R                  ACDFG
6 6                                 ACDEFG
6 7              R  Pl Pl R        AC
6 8                       R        ACDEFG
6 9                    S  R        ACDF
7 0       Ph PhS                   ABC
7 1 S                              BC
7 2       S  PhS                   AB
7 3       S  S                     ABC
7 4 R        R                     BC
7 5 R  S  S                        AC
7 6       PhS PhS                  AC
7 7                                ABC
7 8       PhS PhS                  ABC
7 9 R  S  S                        ABC
8 0                          R  R  ABCDEF
8 1 S          R  Pl Pl R        BC
8 2       R              R        ABDEG
8 3                    R  S        ABCDG
8 4 Pl            Pl S             BCFG
8 5    R             R           ACDFG
8 6    R                         ACDEFG
8 7              R  Pl Pl R        ABC
8 8                                ABCDEFG
8 9                    R           ABCDFG
9 0                             R  ABCDF
9 1 S          R     Pl R        BC
9 2       S           R     R     ABD
9 3                    R           ABCDG
9 4 M             M                BCFG
9 5    R                           ACDFG
9 6    S                        R  ACDF
9 7              R     S  R        ABC
9 8                             R  ABCDFG
9 9                                ABCDFG
'! !

!FoldingDigitAE class methodsFor: 'class initialization' stamp: 'm3r
buildSomeToSomeTransitions
    | index sequences statics |
    #(2 3 4 5 6 8 9) do:
```

```
        [:number | index _ number + 1.
        sequences _ TransitionsTable at: index.
        statics _ (sequences at: 1) copyWithout: #G.
        TransitionsTable at: (12 * index) put: {statics. #spinG}].

    sequences _ TransitionsTable at: 0 + 1.
    TransitionsTable at: 12 "000" put: (sequences copyWith: #diagonalSpinE).
    sequences _ TransitionsTable at: 1 + 1.
    TransitionsTable at: 24 "101" put: (sequences copyWith: #wideRotateC).
    sequences _ TransitionsTable at: 7 + 1.
    TransitionsTable at: 96 "707" put: (sequences copyWith: #rotateAndPushC).




[ ]

!FoldingDigitAE class methodsFor: 'class initialization' stamp: 'm3r
buildTransitionsTable
    TransitionsTable _ Array new: 11 * 12.
    self
        readInTransitionsData;
        buildSameToSameTransitions.
! !

!FoldingDigitAE class methodsFor: 'class initialization' stamp: 'm3r
ensure
    TransitionsTable ifNil:
        [super ensure.
        self buildTransitionsTable].! !

!FoldingDigitAE class methodsFor: 'class initialization' stamp: 'm3r
initializeActions
        "AnimationElement #rebuildAllActions."
    super initializeActions.
    ^self
        addAction: #colorChanged for: #color;
        actions! !


FDDisplayProfile removeSelector: #setMovements:!
FDDisplayProfile removeSelector: #movementDataAt:!
FDDisplayProfile removeSelector: #sequenceAt:!
FDDisplayProfile removeSelector: #setExtentFrom:!
FDDisplayProfile removeSelector: #forms!
FDDisplayProfile removeSelector: #forms:!
FDDisplayProfileFactory removeSelector: #calcStructureUsingConstrainedHeightFor
FDDisplayProfileFactory removeSelector: #initialize!
FDDisplayProfileFactory removeSelector: #recalcExtent!
FDDisplayProfileFactory removeSelector: #storeSequencesFor:offsets:forms:!
FDDisplayProfileFactory removeSelector: #setDisplayProfile:!
FDDisplayProfileFactory removeSelector: #calcSegment!
FDDisplayProfileFactory removeSelector: #storeMovementsFor:offsets:forms:!
FDDisplayProfileFactory removeSelector: #storeUniqueRotations:!
FDDisplayProfileFactory removeSelector: #calcStructureUsingConstrainedWidthFor
FDDisplayProfileFactory removeSelector: #calcMovementData!
FDDisplayProfileFactory removeSelector: #calcStructure!
FDDisplayProfileFactory removeSelector: #defaultSegmentFillRatio!
FDDisplayProfileFactory removeSelector: #smoothingScale:!
FDDisplayProfileFactory removeSelector: #storeUniqueSpins:!
FDSegmentAE removeSelector: #setMovementDataFrom:!
FDSegmentAE removeSelector: #setSequenceFrom:!
FDSegmentAE removeSelector: #displayProfile!
FDSegmentAE removeSelector: #movementDataFrom:!
FDSegmentAE removeSelector: #positionMove:using:!
FDSegmentAE removeSelector: #boundsChanged!
FDSegmentAE removeSelector: #setMovementData:!
FDSegmentAE removeSelector: #setBoundsFrom:!
FoldingDigitAE removeSelector: #updateSegments!
FoldingDigitAE removeSelector: #displayProfile!
FoldingDigitAE removeSelector: #selfValueChangedFrom:!
FoldingDigitAE removeSelector: #segmentNames!
FoldingDigitAE removeSelector: #assignMovements:to:!
FoldingDigitAE removeSelector: #resetDisplayProfile!
FoldingDigitAE class removeSelector: #movementDataAt:!
FoldingDigitAE class removeSelector: #readInMoveTypeFrom:!
FoldingDigitAE class removeSelector: #movementsData!
FoldingDigitAE class removeSelector: #readInMovementLineFrom:!
FoldingDigitAE class removeSelector: #readInMovements!
FoldingDigitAE class removeSelector: #readInSegmentsMovementsFrom:!
FoldingDigitAE class removeSelector: #buildMovementsTable!
FoldingDigitAE class removeSelector: #readInMovementTransitionFrom:!
FoldingDigitAE class removeSelector: #transitionDataAt:!
FoldingDigitAE class removeSelector: #readInSegmentMovementsFrom:!
FoldingDigitAE class removeSelector: #readInMovesTypeFrom:!
```